eMotion 交流伺服馬達實驗手冊

俊原科技技術手冊

Version 1.0

中華民國九十二年十月

eMotion 交流伺服馬達實驗手冊

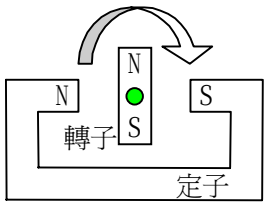
1、電動機原理.....3
2、工作原理.....17
3、實驗規劃.....20
4、實驗項目.....23
5、安裝程序.....24
6、連線程序.....25
7、執行程序.....26
7-1 Telnet 文字模式程式執行步驟.....26
7-2 Matlab/Simulink 環境下程式執行步.....27
8、資料儲存.....28
9、實驗說明.....30
9-1 開環路電壓控制.....31
9-2 開環路定位控制.....41
9-2 閉迴路扭力控制.....44
9-2 閉迴路速度控制.....47
9-2 閉迴路位置控制.....50
9-2 加減速控制與前饋補償.....53
10、模組程式說明.....56
11、Simulink 程式方塊參數設定說明.....77
11-1 Simulink 程式方塊的參數設定範例.....77
11-2 交流馬達控制方塊的參數設定說.....83
12、系統安裝操作說明.....87

1. 電動機原理

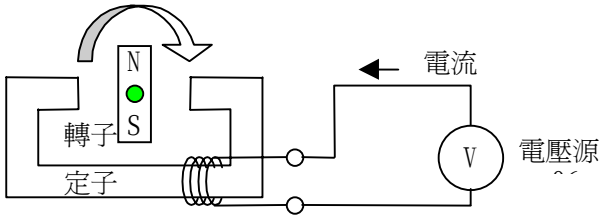
馬達又叫電動機，是利用電場和磁場的交互作用，將電訊號轉換成機械運動。

基本結構

原始構想非常簡單，只是兩組磁鐵擺在一起，其中一個固定不動，叫做定子；另一個中心固定而可以自由旋轉，就叫做轉子。由於其中磁力的影響，就會造成轉子的旋轉運動，如下圖所示：



如果兩組磁鐵都是永久磁鐵，就沒有什麼變化，也不能控制了。但是只要將其中一組換成電磁鐵，那麼利用電流的變化，就可以控制轉子做不同的運動。

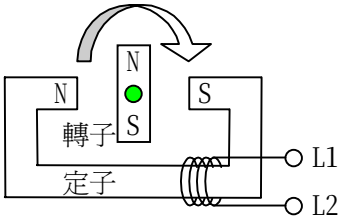


例如上圖以電壓源來驅動定子線圈，由於線圈內組的緣故，電壓越高，電流越大，磁場也越強。因此

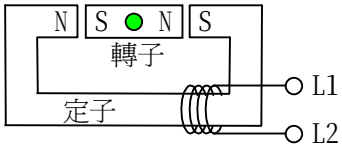
1.由電流的大小改變定子磁場，就可以控制轉子的轉動扭力。
2.由電流的方向改變定子磁場，就可以控制轉子的轉動方向。

這是由於兩個磁場擺成相互垂直，因為同性相斥、異性相吸的結果，會在轉子上產生扭力而迫使其旋轉。但是一旦轉子旋轉 90 度而和定子平行時，轉子就會被磁場鎖住，再強的電流都沒法讓其繼續旋轉了。

很明顯的，從基本結構中，我們立刻可以延伸出下面的兩種控制結構：



扭力控制結構



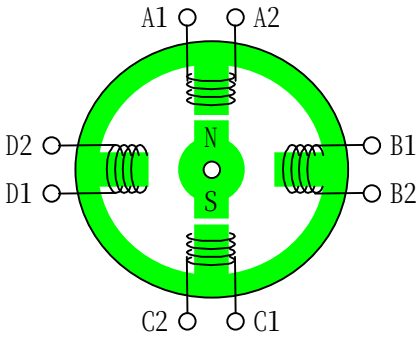
位置控制結構

轉子和定子磁場軸向垂直時就是扭力控制結構，，其扭力大小和方向可以由外加電流的大小和方向來控制。一般的直流馬達和交流馬達都屬於這種結構。

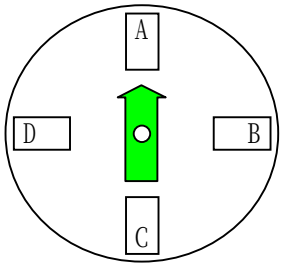
而另一方面，轉子和定子磁場軸向平行時就是位置控制結構，其轉子的位置被定子吸住而固鎖不變，必須經由定子磁場的轉變才有機會改變轉子的位置，步進馬達就是屬於這種控制結構。

步進馬達

在位置控制結構中，馬達轉子被定子磁場鎖住，是完全不能旋轉的。必須多放幾組線圈，再依序控制定子線圈的開關狀態，讓不同角度的定子磁場依序作動，進而導引轉子的轉角，馬達就可以順利的一步步旋轉，這就是步進馬達。

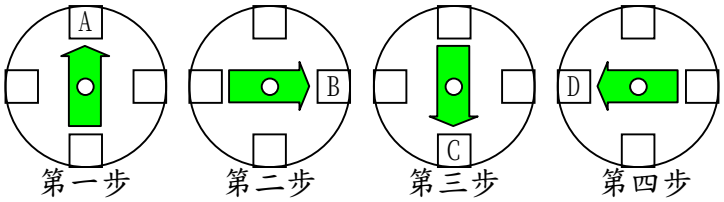


結構示意圖



控制示意圖

上圖就是最基本的步進馬達架構，轉子還是由永久磁鐵製造的固定磁場，但是定子已經換成四組電磁鐵線圈。現在我們對四組線圈分別控制，每次只選擇其中一組的線圈動作；當四組線圈依序輪流動作時，步進馬達就轉起來了。

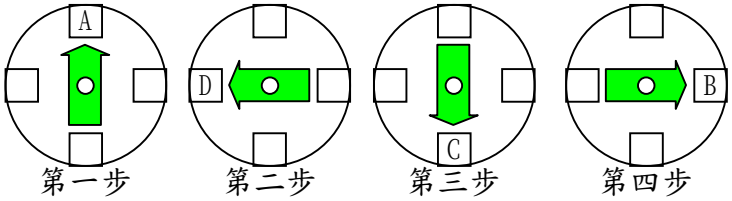


這樣的四組線圈控制，都只是邏輯式的 ON/OFF 控制，非常適合用數位邏輯電路或是微電腦來控制，這也是步進馬達在微型電動機市場非常暢銷的原因。像是磁碟機、印表機和掃描器所用的位移馬達都是採用步進馬達。

步進馬達雖然以位置控制為主，但是一樣可以控制速度和加速度。原因很簡單，位置的變化就是速度，而速度的變化就是加速度。因此

1. 控制定子波形的輸出頻率，就可以控制步進馬達的轉動速度；
2. 控制定子波形的輸出順序，就可以控制步進馬達的正反轉；
3. 當定子波形固定不動時，步進馬達就停止在某一定點，這就是位置控制了。

所以當反轉時，我們只要依序送出下面的定子控制即可：

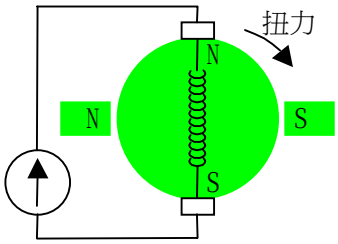


像步進馬達這種位置控制結構的馬達，在運動停止時還要持續供電，能量上非常浪費也很容易發熱，所以不太適用於功率較大的馬達控制。但是在微型馬達上，整體能量耗用並不明顯，效率也沒那麼重要，這時步進馬達簡單好用的特性，就提供了市場競爭的絕對優勢。

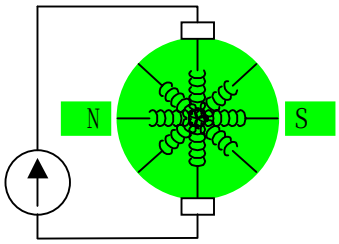
直流馬達

直流馬達的結構和步進馬達完全不同，是屬於轉子和定子磁場垂直的扭力控制結構。當轉子和定子磁場相互垂直時，就會產生扭力而讓轉子旋轉，但是一旦旋轉到 90°，使得兩個磁場平行時，馬達就會失去扭力而停止了轉動。

為了使馬達持續不停的旋轉，直流馬達中設計了很特殊的電樞結構，讓轉子不論旋轉到任何角度，轉子磁場的方向都和定子磁場永遠保持垂直。也因為兩者間永遠能保持垂直，馬達也就能不停的轉下去了。



單一線圈狀態

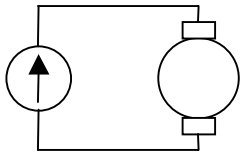


電樞示意圖

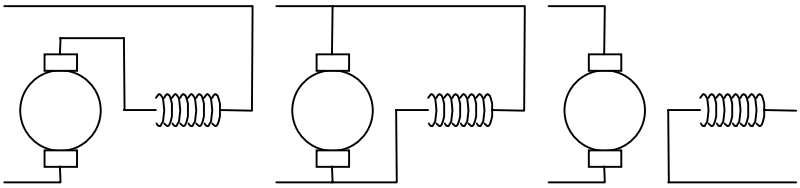
電樞的細部結構說明如下：

1. 在轉子中繞上線圈而建立電磁鐵，一旦通過電流後就能建立和定子磁場垂直的轉子磁場。
2. 比較特殊的是，轉子線圈和外部電源之間的連接面並不是固定式的，而是以電刷在轉子的上下兩端加以滑動連接。只要轉子轉開一些角度，電刷就無法繼續接觸線圈，電流自然不通而轉子磁場也就消失不見了。
3. 接著在轉子上，以不同角度繞上很多組線圈；由於電刷連接的關係，只有唯一的一組線圈(和定子垂直者)能夠導通，並建立轉子磁場。
4. 如此一來，無論轉子轉到任何角度，轉子線圈造成的轉子磁場永遠能和定子磁場垂直，扭力永遠維持，而馬達也就可以不停的旋轉了。

由於電樞和電刷結構是直流馬達中極具革命性的理念，也非常的重要，也難怪簡化之後的直流馬達符號，根本就是一個電樞加兩個電刷了。所以叫「直流」馬達，是因為用直流電源即可驅動。


直流馬達符號

在直流馬達中，定子的作用非常單純，只是維持一個固定的定子磁場而已。如果定子是永久磁鐵，就是永磁式直流馬達。如果定子是電磁鐵，就要根據定子線圈和轉子線圈之間的連接方式，而分成串激式、並激式和分激式三類的直流馬達。


串激式直流馬達　並激式直流馬達　分激式直流馬達

串激式、並激式和分激式直流馬達一般都是作為動力馬達用，可控制的範圍有限。而高精度的的伺服馬達則多採用永磁式的直流馬達，控制效果較

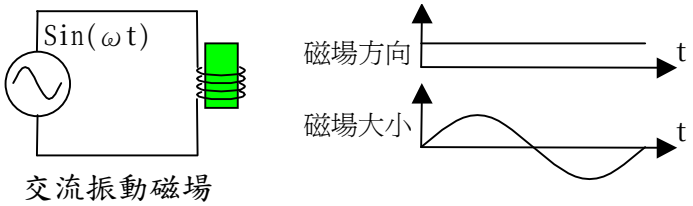
佳。直流馬達的基本結構是扭力控制，但是經由回授控制的處理，一樣的可以控制速度和位置。因為

1. 扭力控制驅動馬達慣量，間接說來就是加速度控制。
2. 加速度的累積就是速度，所以控制加速度就可以間接改變馬達的速度。
3. 速度的累積就是位置，所以控制速度就可以間接控制馬達的位置。

但是因為各項因素的影響都是間接的，很容易受到環境變化和外界干擾的影響，所以閉環路的回授控制是必要的處理步驟。

旋轉磁場

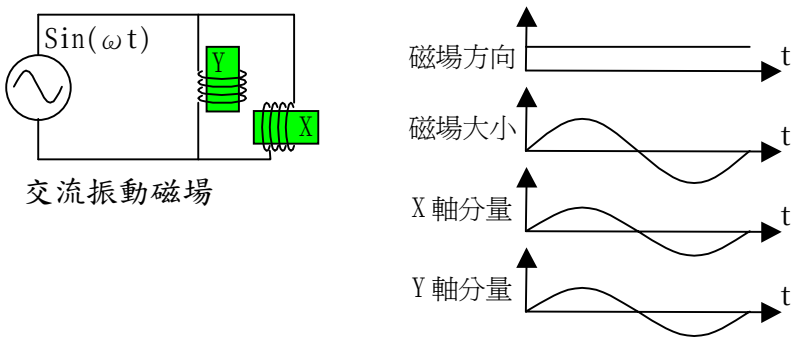
交流馬達的構想和直流馬達一樣，都是建立一組相互垂直的轉子和定子磁場，產生扭力而讓轉子持續的旋轉。為了讓轉子旋轉中的磁場還能維持相互垂直的狀態，直流馬達針對「轉子」而設計了電樞的的架構，而交流馬達則針對「定子」而設計了特殊的旋轉磁場。



交流振動磁場

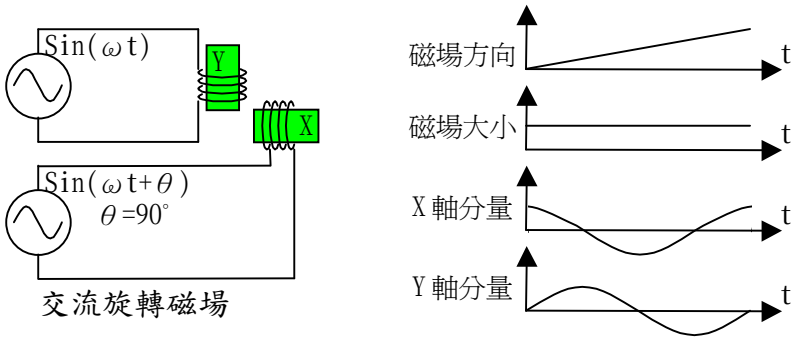
首先要確認的是，單一組線圈是絕對不能構成旋轉磁場的。在上圖中，將交流電流加在線圈兩端時，只會造成「振動」磁場，磁場方向本身是固定不變的。

一旦用交流電源控制相互垂直的兩組線圈時，磁場的方向就有機會改變了。但是交流電源本身也需要做一些不同的處理。在下圖中，我們用同一組交流電源驅動兩組線圈。



交流振動磁場

很清楚的，雖然我們用了 XY 軸的兩組磁場，但是因為兩組線圈使用同一組交流電源，向量組合後的綜合磁場雖然轉到 45°方向，磁場方向還是不變，只是磁場大小變化的振動磁場而已。但是只要將 XY 軸電源換成頻率相同、相位相差 90°的兩組交流電源後，情況就立刻改變了。



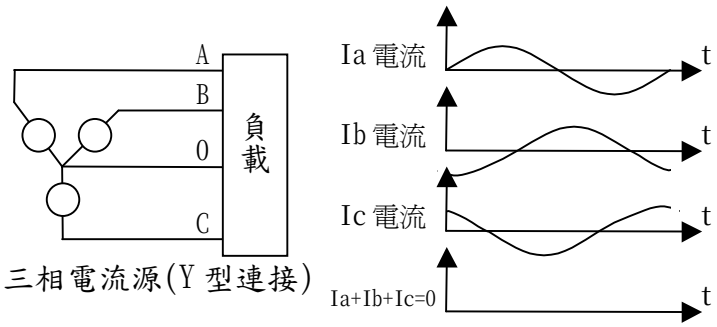
由 XY 軸兩組交流磁場，合成磁場大小固定不變、但角度逐漸增加的旋轉磁場。請特別注意：旋轉磁場雖然接受高頻的交流電源驅動，但是其磁場大小固定不變，完全沒有高頻振動問題，絕對不是「振動」磁場。上面的範例是雙相電源型式的旋轉磁場，我們可以推衍出三相電源型式的旋轉磁場結構：

1. 排列三組電磁鐵，每組磁鐵的機械角度相差120°。
2. 每組磁鐵由獨立的交流電源驅動。這三組交流電源的頻率相同，但是相位各相差120°。
3. 由這三組磁場合成的綜合磁場，就變成大小固定，但是角度逐漸增加的旋轉磁場。

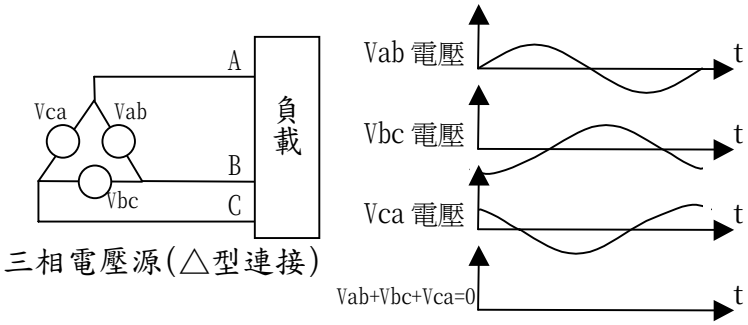
為什麼要由兩相電源換成三相電源？很簡單，因為三相電源的總和永遠為零：

$$\sin(\omega t)+\sin(\omega t+120^\circ)+\sin(\omega t+240^\circ)=0$$

造成三相電源的傳輸效率最好！如果我們將三相的交流電流源做 Y 型連接時，因為 Ia+Ib+Ic=0，所以根本不需要第四根線，只要三根電纜就可以傳輸了。



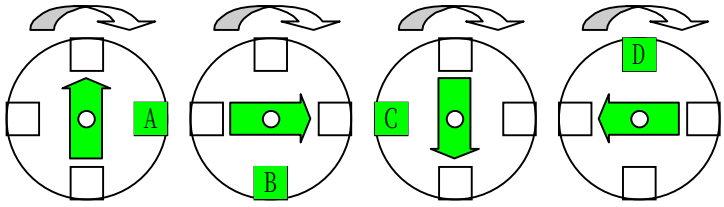
依此類推，如果我們將三相的交流電壓源做△型連接時，因為Vab+Vbc+Vca=0，一樣不需要第四根線，只要三根電纜就可以傳輸。



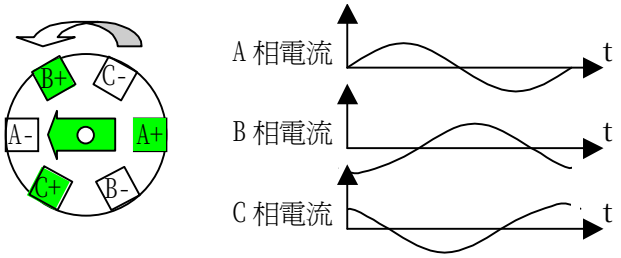
因為傳輸上的效率最佳，現在所有長程傳輸的市電都是採用三相交流電源。因此三相交流電源是我們最容易取得的多相電源，當然交流馬達也多以三相電源為主了。

交流馬達

交流馬達的構想，就是將定子設計成旋轉磁場，當轉子轉動時，定子磁場也跟著旋轉而仍和轉子磁場保持垂直；這樣一來，馬達就能持續保持扭力而繼續旋轉了。



當然用上圖中步進馬達的結構，只要改變定子磁場的控制順序，也可以辦到馬達旋轉的需求。但是因為在扭力轉換的過程之間並不平順，馬達容易抖動，並不是理想的解決方案。交流馬達的定子設計，還是以交流電源合成的旋轉磁場較佳。這也是這種設計稱為「交流」馬達的緣故。



上圖中的交流馬達結構，轉子為永久磁鐵，定子是三相電流驅動的三組線圈所構成的旋轉磁場。當交流電流變化一個週期(360°)時，轉子角度也剛

好轉動一圈(360°)。馬達規格上有一項參數叫「極數」，簡單的說就是轉子N極和S極的總數。以上圖為例，這馬達的極數為2，叫做雙極馬達；當馬達極數=2時：

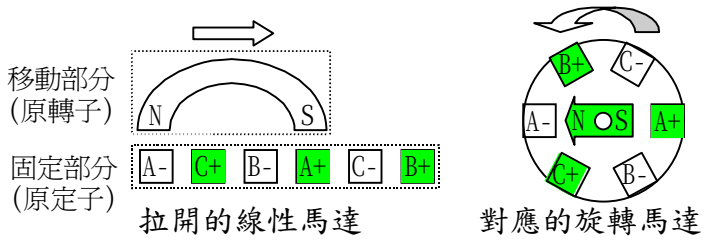
1. 轉子有一對(N/S)極，定子有三對(A/B/C)電磁鐵繞組。
2. 機械角(馬達轉角)和電氣角(電源向量角)的比例爲1:1。
3. 交流電源的一個週期相當於馬達轉一圈。

線性馬達

線性馬達就是把旋轉式的馬達拉成直線而已。其實，直線和圓只是一體的兩面，換算原則就是：

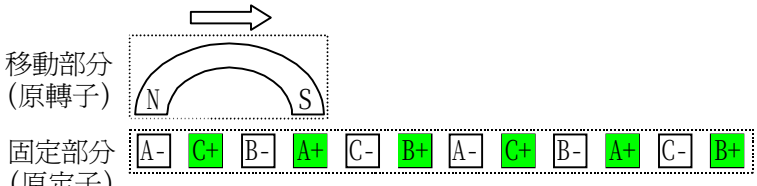
1. 圓弧的半徑拉成無限大，就等效於直線。
2. 直線段就等效於半徑無限大的一段圓弧。

所以將前面的交流馬達想成是一個黏土，從轉子中心剖開攤平，就形成下面的線性馬達：



拉開的線性馬達　　對應的旋轉馬達

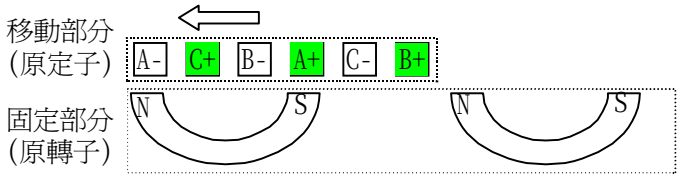
只是線性馬達有移動範圍的限制，不像旋轉馬達可以無限制的旋轉。當要延長移動範圍時，就必須將定子部分多複製幾份，舉例如下：



延長的線性馬達

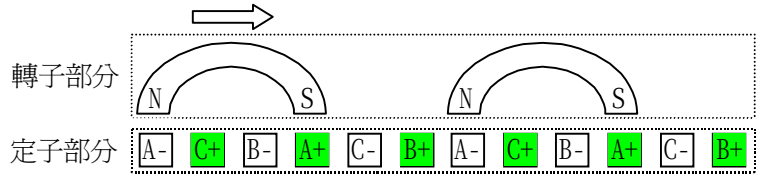
像這樣子的線性馬達，其移動平台部分並無控制，所有的電磁鐵都在固定部分控制。其優點是不需將接線配到移動平台上，缺點是當移動範圍增大時線圈的繞組和配線會越來越麻煩。其實移動平台和固定部分呈現相對的運動，我們可以將其顛倒配置，就成為另一種的線性馬達。

這種線性馬達的控制線圈(原定子部分)放在移動平台上，而固定部分只是單純的永久磁鐵(原轉子部分)。當移動範圍加大時，只要增加永久磁鐵的個數，不會增加控制和配線的困難度，系統安裝就可以簡化許多。

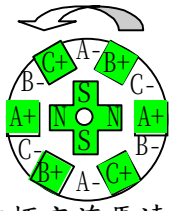


多極馬達

馬達極數越多，精度越高，也越能提供較大的扭力。如何構成多極馬達呢？我們可以將雙極的交流馬達攤平成線性馬達，再將機組複製一份，就形成下面的設計：



接著將攤平的馬達再捲回旋轉馬達的型式，就構成下圖中四極的交流馬達。其中的定子是兩對的(N/S)極的永久磁鐵；而定子為六對的(A/B/C)電磁鐵繞組。必須注意的是，只需要三相的交流電源驅動(而不是六相)，同相的電源間是用串聯或並聯在一起來處理。如此一來，當交流電流變化一個週期(360°)時，轉子角度也剛好轉完半圈(180°)。



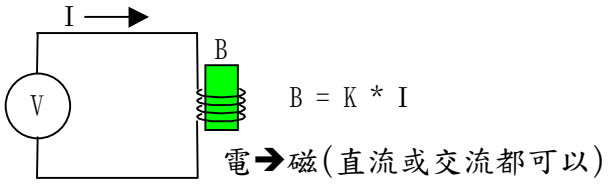
四極交流馬達

依此類推，當馬達極數從雙極增加為 N 極(N 必須是偶數)時，其特性如下：

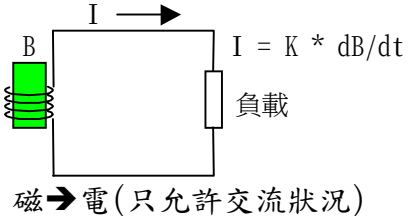
1. 轉子有(N/2)對(N/S)極，定子有(N/2*3)對(A/B/C)電磁鐵繞組。
2. 定子只用一組三相電源驅動，所有重複繞組都用並聯或串聯方式結合在一起。
3. 機械角(馬達轉角)和電氣角(電源向量角)的比例爲2:N。
4. 交流電源的一個週期相當於馬達轉(2/N)圈，所以控制精度增加爲(N/2)倍，而扭力也增加爲(N/2)倍。

變壓器

馬達要產生扭力，靠得是兩組磁鐵之間的磁力作用。而馬達要控制扭力，就必須靠著將電場變成磁場的電磁鐵。

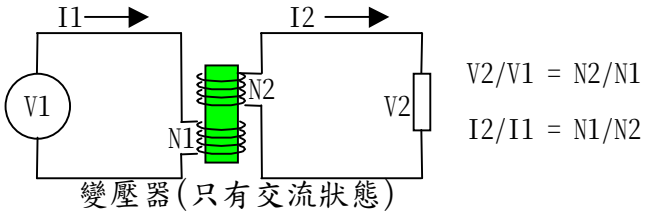


電磁鐵是由電生磁的基本手段，其產生的磁通量(B)和電流(I)成比例，而這比例值(K)和線圈截面積、繞線圈數、以及鐵心特性都有直接關係。要注意的是，由電生磁的作用和電源種類無關，無論是直流電和交流電都具有同樣的特性和公式。



類似結構也可以將磁變回電，其電流(I)和磁場微分量(dB/dt)成比例。因此只有交流磁場才能變回電場，而直流磁場是變不回來的。其中比例值(K)和線圈截面積、繞線圈數都有直接關係。

將兩者整合在一起就形成變壓器。變壓器分成初級圈(輸入端)和次級圈(輸出端)兩組線圈。如果次級圈和初級圈的繞線圈數比為(N2/N1)，則其電壓比(V2/V1)和其成正比，而其電流比(I2/I1)和其成反比。條件是變壓器本身無能量耗損，即 V1*I1=V2*I2。

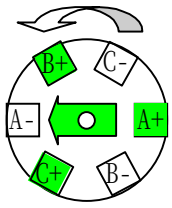


變壓器的輸入是電能、輸出也是電能，並不能提供能量轉換的作用，其特性為：

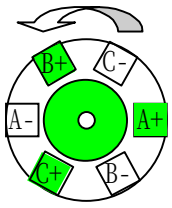
1. 在能量不變的條件下，改變輸入和輸出之間的電壓和電流。
2. 輸入輸出之間可以不共地(Ground)，提供訊號隔離的作用。
3. 利用空間傳遞電能，而不需經過真正的配線連接。
4. 只適用於交流電源，不適用於直流電源。

感應馬達

前面所提的交流馬達都是交流同步馬達，其定子是三對電磁鐵構成的旋轉磁場，而轉子為永久磁鐵。為了降低製造成本，可以將轉子換成電磁鐵結構，就構成了交流感應馬達。



交流同步馬達



交流感應馬達

如果將感應馬達視為變壓器，那麼定子就是變壓器的初級圈，而轉子是次級圈；不需要配線連接，就可以將電能傳到轉子而產生電磁鐵效應。這電磁鐵所產生的轉子磁場就可以取代永久磁鐵，而讓交流馬達持續的運轉。為了造成變壓器效應，交流感應馬達有幾個特色：

1. 當然只能適用於交流電，才能透過變壓器效應傳遞電能。
2. 如果轉子轉速和定子磁場的旋轉速度完全一致；那麼就轉子而言，其周遭的磁場相當於固定不變。在沒有交流磁場的狀況下，是不可能產生變壓器效應而傳遞電能的。
3. 換句話說，轉子的轉速一定要和定子磁場的旋轉速度不同，其間滑差(Slip)才會造成磁場的變化，才能將磁場變化再轉成電能傳遞轉子上。
4. 轉移至轉子上的電能都會消耗在產生電磁鐵的工作上，是馬達中額外的能量消耗。

因為靠著電磁感應來工作，所以稱之為「交流感應馬達」。因為轉子變壓器的架構很像老鼠籠，也俗稱為「鼠籠式馬達」。因為感應馬達必須有滑差所以轉速和定子磁場一定不同步；相反的，用永久磁鐵做轉子的馬達，轉速就能和定子磁場完全同步，所以就稱為「交流同步馬達」。

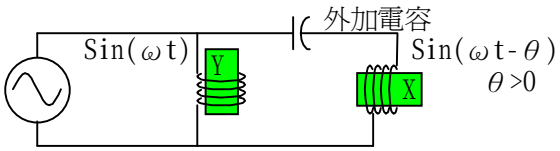
感應馬達因為在能量轉換上的一些額外消耗，所以效率較低；這些額外的耗能又造成散熱問題，所以馬達體積也較大；所以在應用上還是以低價位的動力馬達市場為主。相對的，精密控制的應用市場還是以交流同步馬達佔多數。

單相交流馬達

在功率較低的應用場合，為了顧及用電的方便性，必須考慮到單相電源的應用環境。由於旋轉磁場最少要有兩相電源，所以如何將單相電源擴張到兩相電源，就是單相交流馬達的主要工作。

理想上來說，兩相交流電源之間必須有 90°的相位差。但是為了降低生產成本，實際上兩相之間只要存在一點相位差，即可造成旋轉磁場，雖然不太均勻，但是馬達仍可順利運轉。

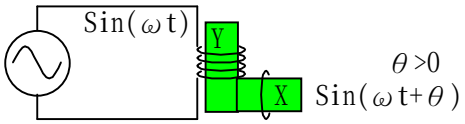
為了造成這一點的相位差，我們可以用電容造成相位延遲，而設計出電容式馬達。



電容式馬達的定子示意圖

電容式馬達的結構還是標準的兩相交流馬達，但是在配線上，經由外加電容的移相作用，X/Y 軸兩組電磁鐵上的電源就有了一些相位差；靠著這一點相位差，馬達就旋轉起來了。

另一種特殊的設計是蔽極式馬達，這是在單相馬達的鐵心上繞上一小圈的短路環，產生變壓器的次級圈效應。但是因為感應出的電流被本身短路了，這個短路電流又產生了新磁場而抵銷一部份原來磁場的作用，而間接造成磁場的相移現象；靠著這一點相位差，馬達就旋轉起來了。



蔽極式馬達的定子示意圖

不論是電容式馬達或是蔽極式馬達，其能量轉換效率都很差，所以只能用在中小功率、低成本的的動力應用上，並不適用於高精密的控制馬達市場。但是因為電源方便成本又低，所以許多家用馬達(像是電風扇)都是用蔽極式或電容式的單相交流馬達。

電動機分類

電動機有許多不同的分類方式，列舉如下。

依控制結構分類，馬達可分為
1. 位置控制結構：步進馬達。
2. 扭力控制結構：直流馬達和交流馬達。

依磁場結構分類，馬達可分為
1. 定向磁場結構：直流馬達(電樞結構)。
2. 旋轉磁場結構：交流馬達和步進馬達。

依定子結構分類，直流馬達可分為
1. 永久磁鐵結構：永磁馬達。

2. 電磁鐵結構：串激馬達、並激馬達、分激馬達。

依轉子結構分類，交流馬達可分為

3. 永久磁鐵結構：同步馬達。
4. 電磁鐵結構：　感應馬達。

依運動方式分類，交流同步馬達可分成

3. 旋轉運動：交流馬達。
4. 直線運動：線性馬達。

依電源結構分類，交流感應馬達可分為

1. 三相電源：三相馬達。
2. 單相電源：單相馬達。

依定子結構分類，單相交流感應馬達可分為

1. 電容分相：電容式馬達。
2. 蔽極分相：蔽極式馬達。

依應用市場分類，馬達可分為

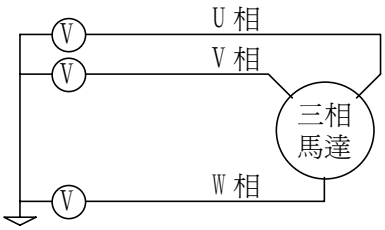
1. 精密控制：直流永磁馬達、交流同步馬達、步進馬達。
2. 動力馬達：交流感應馬達、直流串激馬達、直流分激馬達。

本書主要以精密控制為主，在後面各章中將逐步說明馬達控制上所需的各項設計技術，尤其著重在精密伺服控制部分。

2. 工作原理

三相馬達

三相馬達的基本接線方式如下：



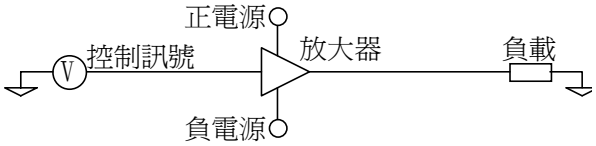
其中：

1.馬達線圈分成 U/V/W 三相，分別由三組獨立的電壓源作驅動。

2.每組電壓源的控制都是完全獨立的。但是為了方便整體的控制，可將其定義成三相交流電源處理，三者的振幅一致、頻率一致、但相位各差 120 度。

功率驅動級

一般而言，從小功率的控制訊號到大功率的負載驅動，都需要經過功率驅動級，圖示如



下：

其中：

1.以功率放大器將控制訊號放大，並用以驅動大功率的馬達負載。

2.放大器的輸出電壓和電流由正負電源所提供，計算公式如下：

A.如果電源電壓為V_S，負載電壓為V_L，而負載電流為I_L。

B.則負載上的功率輸出為$W_L=V_L*I_L$。

C.而電源所提供的功率輸入為$W_S=V_S*I_L$。

D.兩者間的功率差值即是放大器上的功率消耗，$W_T=(V_S-V_L)*I_L$。

3.放大器本身的功率消耗(W_T)不但是完全浪費的，還會轉會成熱能而產生嚴重的散熱需求。因此功率放大器的設計原則，是以最低的功率消耗為設計目標。

如果控制訊號是正弦波型式的交流訊號，那麼負載上的平均振幅就是其最大振幅的 0.7 倍。因為電源電壓必須超過負載的最大電壓，因此以統計值而言，$V_L>=0.7V_S$。所以在功率放大器中會產生 30%以上的功率消耗。例如：

1.對於 1kW 的電源，可驅動 700W 的馬達負載，而產生 300W 的功率消耗。

2.對於 10kW 的電源，可驅動 7KW 的馬達負載，而產生 3000W 的功率消耗。

3.一支電子實驗用烙鐵耗電約 20W，一般電鍋或電爐的耗電約 800W。

換句話說，驅動 1Hp 的馬達時功率驅動級中會產生 300W 的耗電，這是 15 支烙鐵所產生的熱能。而驅動 10Hp 的馬達時功率驅動級中會產生 3000W 的耗電，這是 4 個電鍋所產生的熱能。

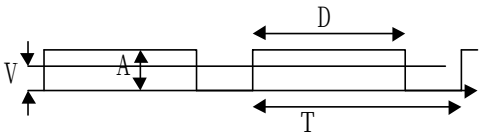
PWM 電壓控制

仔細檢討功率驅動級的設計，可發現最低的功率消耗點發生於：

1.最低電壓時，這時V_L=0 且I_L=0，所以$W_T=(V_S-V_L)*I_L=0$。
2.最高電壓時，這時$V_L=V_S$，所以不論負載電流多大，$W_T=(V_S-V_L)*I_L=0$。

如果功率驅動級只准輸出最高電壓和最低電壓，就能得到最低的功率消耗。最高電壓為 ON，而最低電壓為 OFF，所以 ON/OFF 型式的切換式功率驅動級具有最佳的功率消耗因素。

脈寬調變技術(Pulse Width Modulation，簡稱 PWM)就是最常用的切換式功率驅動級。

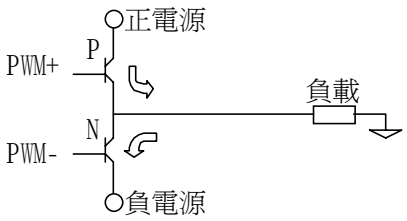


示意圖如下：
其中：

1.PWM 波形的輸出只有 ON/OFF 兩種狀態，其中 ON 的電壓(A)為其最大電壓值，而 OFF 的電壓為 0。
2.如果其週期為 T，ON 的時間為 D，則 PWM 所產生的平均電壓為 V=(D/T)*A。換句話說，可以產生 0~A 之間的所有電壓範圍。
3.但是如果切換的頻率過低，會造成馬達輸出時的震動現象，或是線圈中的高頻噪音。一般而言，合理的切換頻率在 4KHz~8KHz 間。
4.馬達的電機特性就是一個低通濾波器，過高的頻率是不會響應的。因此只要 PWM 的切換頻率夠高，其特性和線性放大器是一致的。

PWM 切換保護

對於單一線圈的負載而言，PWM 切換電路的電路方塊圖如下：



其中：

1.以兩組電晶體來分別控制正負電源，其中的電晶體依據功率需求可選擇功率電晶體、MOSFET 或是 IGBT。不論哪種選擇，其控制原理都一致。

2.將 PWM 訊號分成正向和反向的兩組控制訊號，PWM+控制正電源的導通與否，而 PWM-控制負電源的導通與否。

3.切換組合包括：

P=OFF/N=OFF：負載和電源隔離。

P=ON /N=OFF：負載連接上正電源。

P=OFF/N=ON ：負載連接上負電源。

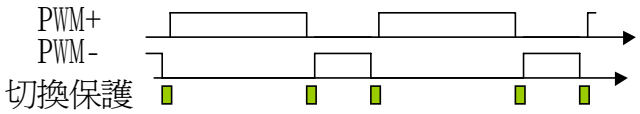
P=ON /N=ON ：正負電源短路，這時功率驅動級將瞬間燒毀。

原則上只要將 PWM+和 PWM-間的邏輯反向，是不可能發生兩者都 ON 的情況。但是考慮功率電晶體(包括電晶體、MOSFET 和 IGBT)的暫態響應，將電晶體 OFF 的反應時間通常 ON 的反應時間稍長，所以在切換的瞬間很可能產生兩者偕導通的瞬間狀態。這時可能有兩種結果：

1.兩者同時導通的時間稍長時，功率驅動級將瞬間燒毀。

2.兩者同時導通的時間非常短，這時功率驅動級雖不至於瞬間燒毀，但是會產生很高的熱量，使得功率驅動級的溫度快速加高，甚至於過熱燒毀。

因此 PWM 控制必須加上切換保護，其示意圖如下：



其中：

1.基本上，PWM-只是 PWM+的反向訊號，兩者都由一組 PWM 訊號控制。

2.但是在 ON/OFF 切換的瞬間，必須加上一段兩者都 OFF 的時間。這段時間的長度可依照功率電晶體的 ON/OFF 特性來設定，在 1us~4us 之間。

PWM 啟動保護

在系統開機時，很不容易保證各個元件的理想啟動程序。因此在系統啟動時必須自動將 PWM 輸出關掉，直到系統完成整個開機程序後，才能再將 PWM 訊號重新啟動。一般而言，啟動程序如下：

1.系統開機後，功率驅動級通常最快啟動。這時 PWM 必須保持在 OFF 狀態，避免系統誤動或短路。

2.之後依次啟動 BIOS、Linux、網路，最後才啟動應用程式。

3.應用程式啟動後，必須依序

A.下載 FPGA，

B.下載 DSP 控制程式，

C.啟動 DSP 控制程式。

在啟動 DSP 控制程式後，才能重新開啟 PWM 的動作。

4.任何時間中只要 DSP 程式停止，就必須立刻關閉 PWM 控制，以避免誤動或短路。

所以 PWM 啟動保護是由 DSP 程式直接控制，其必要程序包括：

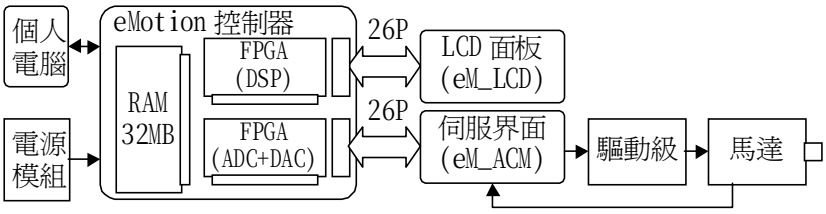
1.FPGA 電路下載完畢並執行工作中。
2.DSP 程式下載完畢並執行工作中。

其中任何一個部分停止都必須立刻關閉 PWM 控制。一般而言，這部分電路都由看護犬(watch-dog)電路管理，而由 DSP 程式負責驅動。

3. 實驗規劃

實驗配置

實驗模組的配置規劃如下：



其中包括：

1.標準的 eMotion 控制器配置，包括電源模組和以網路連線的個人電腦。

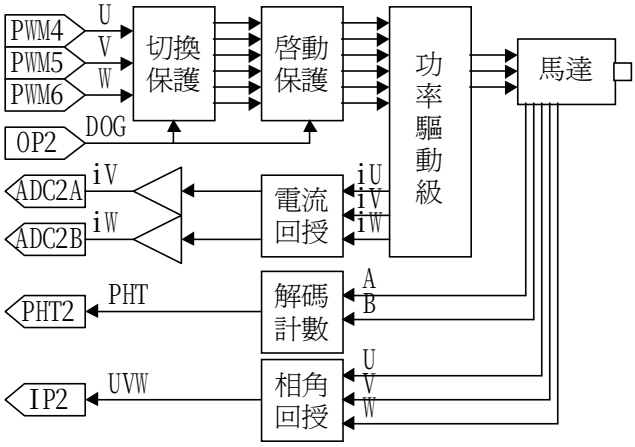
2.兩組 26P 外接模組中，

X 模組：接上 LCD 面板，做為實驗波形的監控顯示用。

Y 模組：接上伺服界面，用以連接驅動級和馬達回授，控制交流伺服馬達。

3.實驗結果以 LCD 模組的示波器和邏輯分析儀功能做觀測。

就實驗中的伺服界面而言，電路方塊圖如下：



其中：

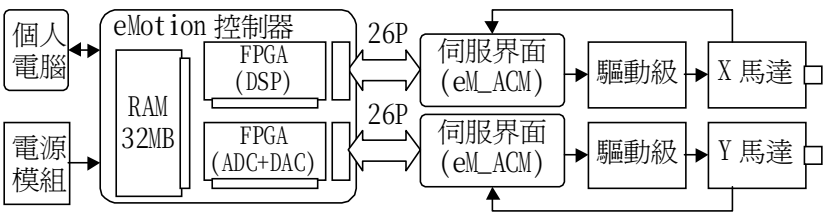
1.由三組 PWM 產生器控制 6 相的 PWM 功率驅動級，驅動交流伺服馬達。

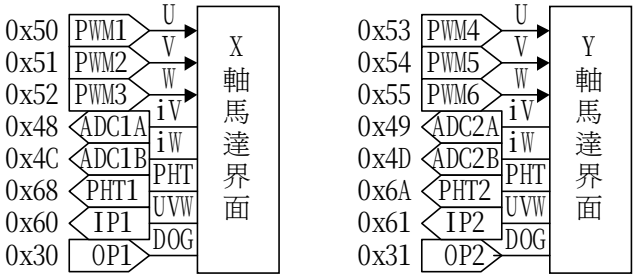
2.實驗時不斷送出方波執行啟動保護，避免程式下載或是馬達停機時的誤動。

3.三相電流只需讀出兩組就可進行三相的電流回授。

4.馬達的角度回授分別做相對角度的 A/B 相回授和絕對角度的 U/V/W 相回授。

這種結構很容易擴張至兩軸馬達的控制，配置規劃如下：





FPGA#2 的電路則換成 testACM2.bit，電路方塊圖如下：

其中：

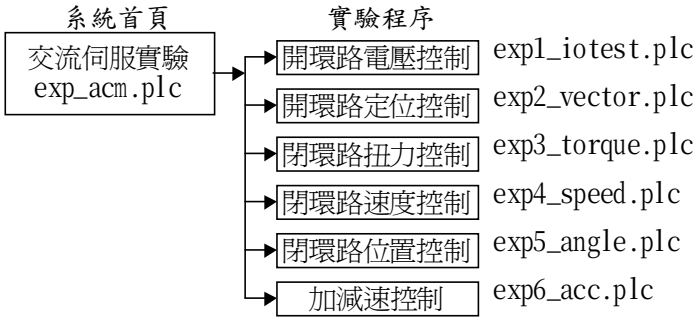
1.兩組 26P 的外接模組中，X 模組接上 X 軸馬達界面，Y 模組接上 X 軸馬達界面。由於沒有接上 LCD 模組，所以無法再使用示波器功能。

2.以六組 PWM 界面分別驅動兩軸馬達，以四組 ADC 界面量測電流回授。

3.以兩組 PHT 界面量測馬達角度回授，以兩組 IP 界面量測馬達相位回授。

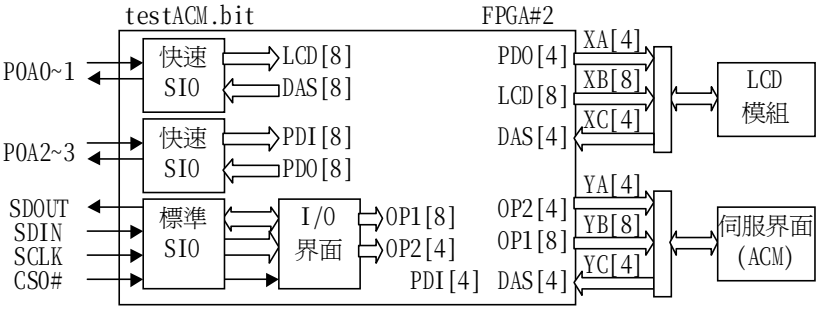
4.以兩組 OP 界面設定功率驅動級的保護參數。

實驗項目



FPGA 界面

在本實驗中，下載到 FPGA#2 中電路檔為 testACM.bit，設計目標是對 LCD 模組和伺服界面模組做連線。電路方塊圖如下：



其中包括：

1.以兩組快速 I/O 對 FPGA#1 做連線，由 FPGA#1 直接控制 LCD。

2.將 XC[4]和 YC[4]的訊號送至邏輯輸入點 DAS[8]，由邏輯分析儀做監控。

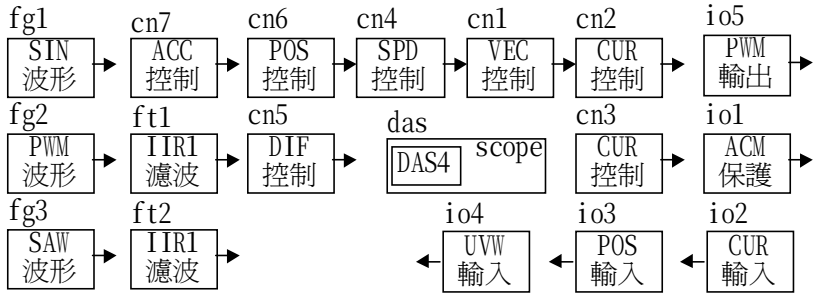
3.YB[8]和 YA[4]由 DSP 核心控制直接控制，對映位址為 OP1(0x30)和 OP2(0x31)。

系統程式設計

系統程式範例：testACM.sys

```
@module    io1   ACM_protect.mod;    //驅動級保護
@module    io2   CUR_input.mod;      //三相電流讀值
@module    io3   POS_input.mod;      //位置和速度讀值
@module    io4   UVW_input.mod;      //相位回授讀值
@module    fg1   SIN_wave.mod;       //弦波產生器
@module    fg2   PWM_wave.mod;       //方波產生器
@module    fg3   SAW_wave.mod;       //鋸齒波產生器
@module    cn7   ACC_control.mod;    //加減速控制器
@module    cn6   POS_control.mod;    //位置控制器
@module    cn5   DIF_control.mod;    //差分控制器
@module    cn4   SPD_control.mod;    //速度控制器
@module    cn3   CUR_control.mod;    //W 相電流控制器
@module    cn2   CUR_control.mod;    //V 相電流控制器
@module    cn1   VEC_control.mod;    //三相分配電路
@module    ft1   IIR1_filter.mod;    //一階濾波器
@module    ft2   IIR1_filter.mod;    //一階濾波器
@module    io5   PWM_output.mod;     //PWM 電壓輸出界面
@module    das   das4.mod;           //示波器資料蒐集
```

為了搭配各個實驗程序，系統程式的方塊圖如下:



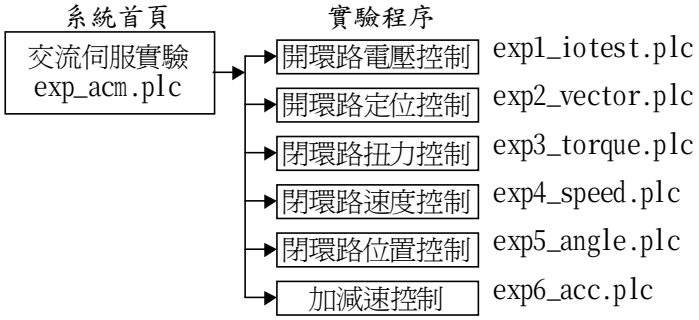
其中包括：

1.三組波型產生器作為測試命令，包括正弦波、方波和鋸齒波。
2.七組控制器執行包括加減速、位置、速度、相位、電流控制器的工作，可依照實驗需求搭配使用。
3.五組輸入輸出界面，執行 PWM 輸出、驅動保護、電流回授、角度回授和相位回授的界面處理。
4.兩組低通濾波器作為電流迴路和速度迴路的模擬使用。
5.DAS4.mod 是示波器專用的配合模組，不論任何實驗都必須加入此模組。

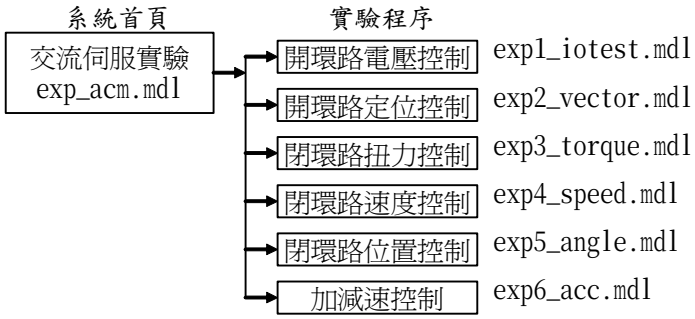
4. 實驗項目

在 eMotion 系統中，內建二種操作模式，一為使用 Telnet 為操作界面的文字模式，二為使用 Matlab/Simulink 為操作界面的視窗模式。所規劃的實驗項目，可以相互比對，茲表列如下：

4.1 在 Telnet 程式中，透過遠端連線程式，以文字模式進行 AC 馬達控制的實驗項目。



4.2 在 Matalb 程式中，以圖形介面進行同樣的實驗。使用者可透過親和力較高的圖形介面，進行線上的參數調校。



5. 安裝程序

5.1 將 eACM8 simulink.zip 中的程式，解壓縮儲存到 **c:\smb\simulink** 的資料夾中。其中包含下列檔案：

檔案	內容說明
*.mod	POS_inDCM.MOD、PWM_outDCM.MOD、ACC_conDCM.MOD、PD_control.MOD 這些檔案是重新 link 出 eMot.mem 所需的模組檔。
*.asm	POS_inDCM.asm、PWM_outDCM.asm、ACC_conDCM.asm、PD_control.asm 是重新組譯模組檔所需的組語 SOURCE 檔。
*.plc	為進行實驗所需的各批次檔。
*.bit	testacm.bit 為下載到 FPGA#2 的 BIT 檔其功能為提供交流伺服控制所需的 PWM dead zone 保護電路，testdsp.bit 則為下載到 FPGA#1 的 DSP 核心檔。
testACM.sys	記載進行 ACM 實驗時所需的模組。
testACM.MEM	連結完成的 DSP 執行檔，用來進行 ACM 實驗。
testACM.var	紀錄 ACM 實驗時的參數對照表。

5.2 將 eACM8.zip 中的程式，解壓縮儲存到 **c:\smb\eACM8** 的資料夾中。其中包含下列檔案：

檔案名稱	內容說明
*.mdl	為使用 Matlab/Simulink 進行實驗所需的 mdl 檔。
*.m	exp_dcm.m 為實驗首頁，eplot 用來繪製實驗數據曲線
eMotion.dat	紀錄 eTTY 所在之目錄和 DSP 的抽樣頻率。
DspBlock.dll	S-function 的動態連結檔,提供即時傳送命令和控制參數到 eTTY 的功能。

5.3 將 Images.zip 解壓儲存到 **c:\smb\Simulink\Images** 的資料夾中，該壓縮檔中有許多圖檔,其功用在正常顯示 Simulink 中,實驗所需各 mdl 檔功能方塊的 S-function block 上的圖示。

6. 連線程序

(1) 將個人電腦與 eMotion 控制器，經由 Telnet 程式界面，進行連結。(詳細 login 的步驟，參考第 12 章系統安裝操作說明)

(2) 在 Telnet 的環境下，將路徑更改到 cd /smb/simulink

(3) 執行 ./eTTY 192.169.1.136

(4) 出現 eM> 提示指令，此時可下達各式控制指令。

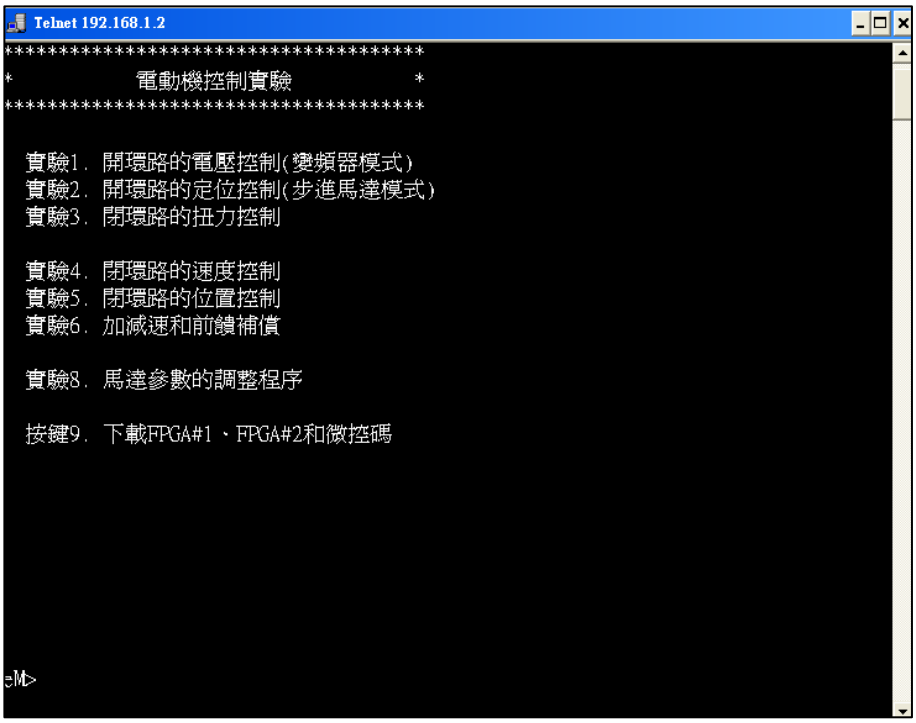
7. 執行程序

7.1 Telnet 文字模式程式執行步驟

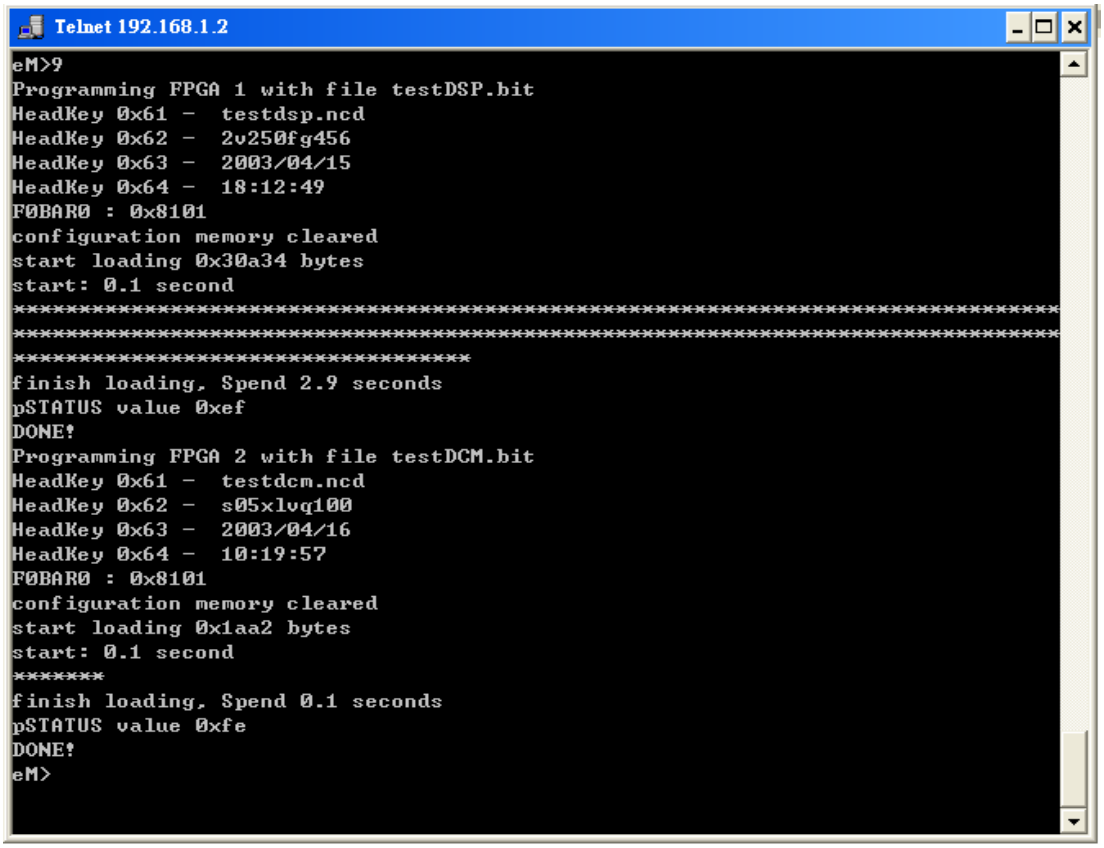
(1) 延續第 4 節中的連線程序，在 eM> 提示符號下執行 prog exp_acm.plc，出現如圖二之實驗選單。

(2) 在圖二所示之選單中，首先選 9 下載*.bit 檔和微控碼，下載完成後將出現如圖三之畫面。

(3) 在圖二所示之選單中，依提示按下 1~5 選擇要進行的實驗，並由 LCD 模組觀察實驗結果(各實驗的詳細說明，參見第 7 節)。



圖一 Telnet 文字模式下之 AC 馬達實驗畫面



圖二 FPGA bit 檔和微控碼下載完成畫面

7.2 Matlab/Simulink 環境下程式執行步驟

(1) 啟動 Matlab 6.1 或以上的版本 (注意!!! 使用高版本的 Matlab/Simulink 後，若有存檔的動作，回到低版本的 Matlab/Simulink 的環境下，將無法開啟原低版本之 *.mdl 檔) 。

(2) 將 Matlab 的路徑，更改到 c:\smb\eACM8。

(3) 執行 exp_acm.m。

(4) 螢幕出現交流伺服馬達控制實驗畫面 (參見圖四)。

(5) 先按 “FPGA 程式下載” 鍵，下載 FPGA 的程式 (*.bit 檔)。

(6) 依序按下實驗 1~6 的按鍵，進行交流伺服馬達的各項實驗(參見圖七~圖十二) 。在執行 Simulink 的程式時，請隨時注意 Telnet 連線程式中的螢幕訊息變化，以檢視 Matlab/Simulink 程式，與 eMotion 控制器的連線情形，與回傳訊息。

注意：關於 DSP 核心的取樣頻率的設定，可以 NOTEPAD 或其他編輯程式開啟後修改 eMotion.dat 的內容。修改第二行中的數字，執行時就可依需求，設定 DSP 核心的工作取樣頻率。

c:\smb\simulink\　　(設定 DSP 核心程式檔的所在資料夾位置)

8　　(設定工作取樣頻率)

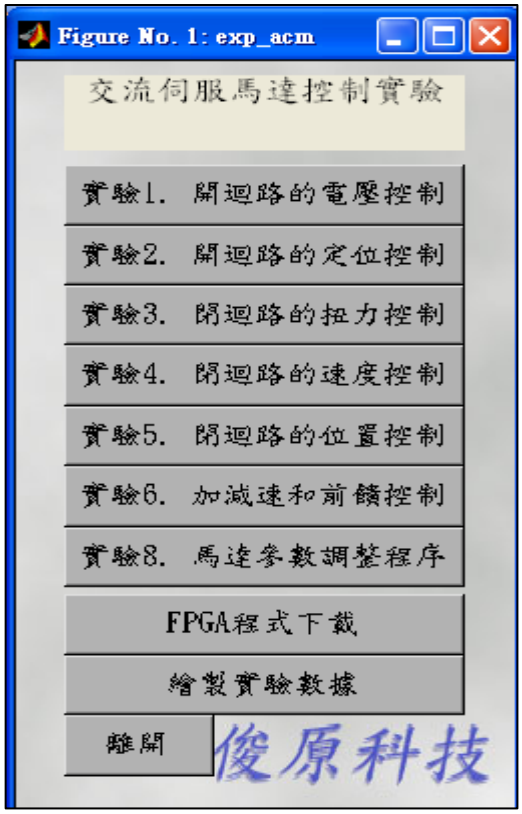


圖 4-1 交流伺服馬達控制之 Simulink 操作界面

8. 資料儲存

(1) 在 Simulink 環境下，執行交流伺服馬達控制的各項實驗。

(2) 經 LCD 模組，檢視資料擷取與顯示情形，並確認無誤。

(3) 先將圖 4-2：trg 打開，顯示如圖 5-1：在 display 打 V 再按 apply 鍵，如圖 5-2 再將在 display 取消打 V 再按 apply 鍵完成記錄，再按圖四：按下繪圖實驗數據，即可顯示下列圖形，在 **c:\smb\simulink** 資料夾中，將出現最新版的 eMot.dsp 資料檔。該檔案紀錄最近在 LCD 模組上，並配合 **c:\smb\eACM8\eplot.m 之繪圖程式**。

(4) 回到圖 4-1 的界面下，按下 ”繪製實驗數據” 按鍵後，即可呈現最新資料圖形。(參考圖 5-3)

(5) 為使資料可供後續應用，請自行將 eMot.dsp 更名後另外儲存。

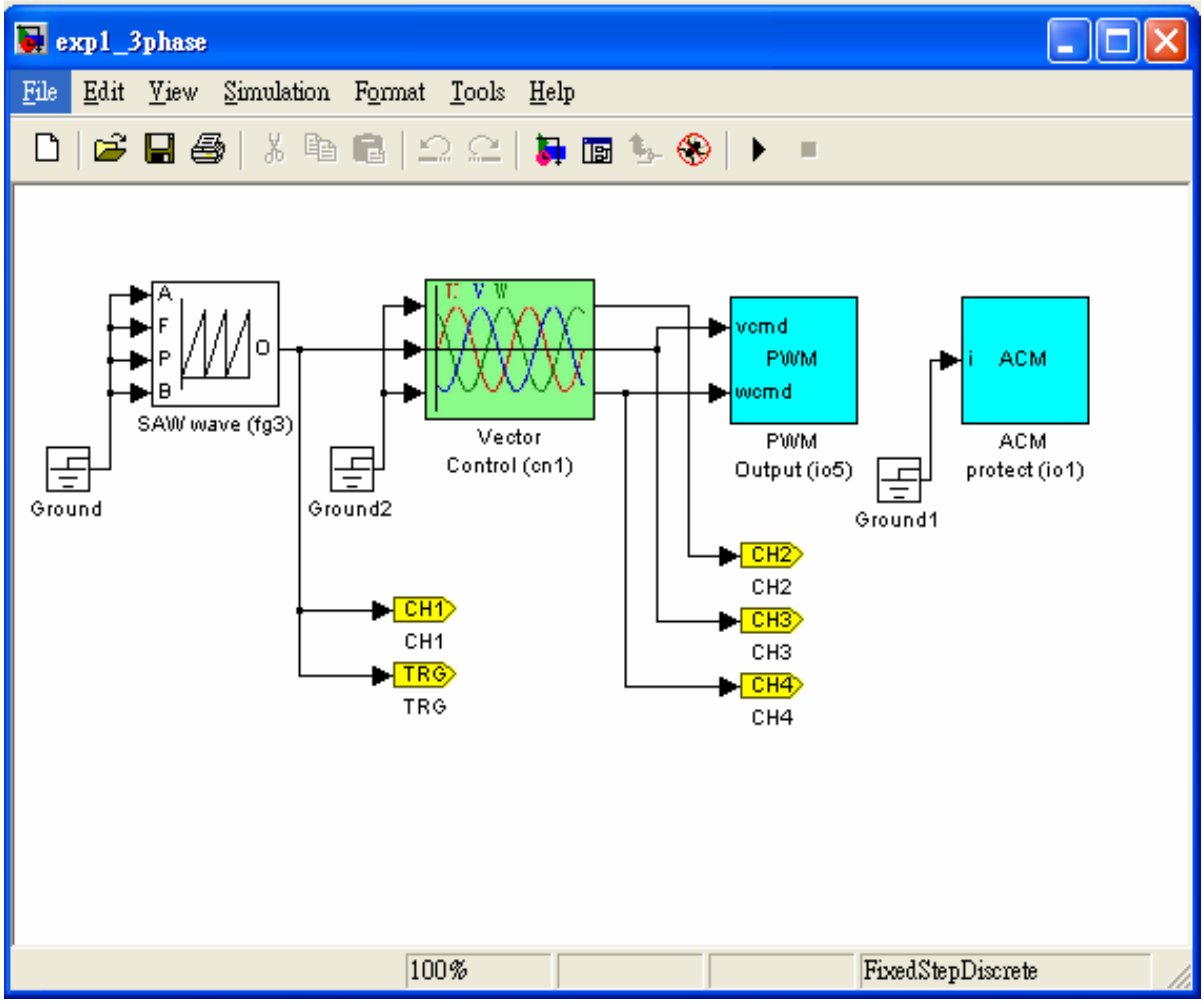


圖 4-2 SIMULINK 環境下實驗操作畫面

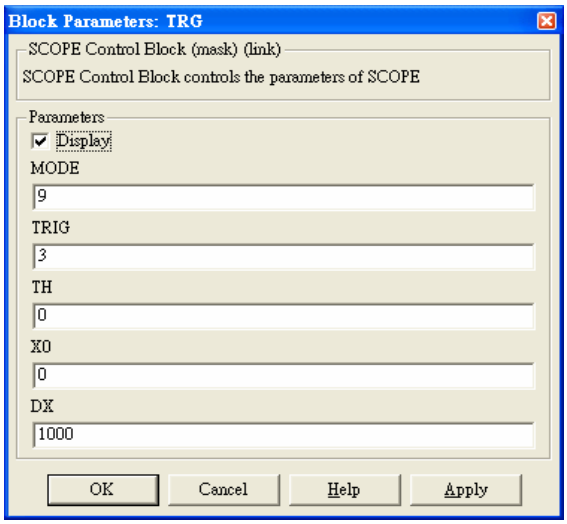


圖 5-1

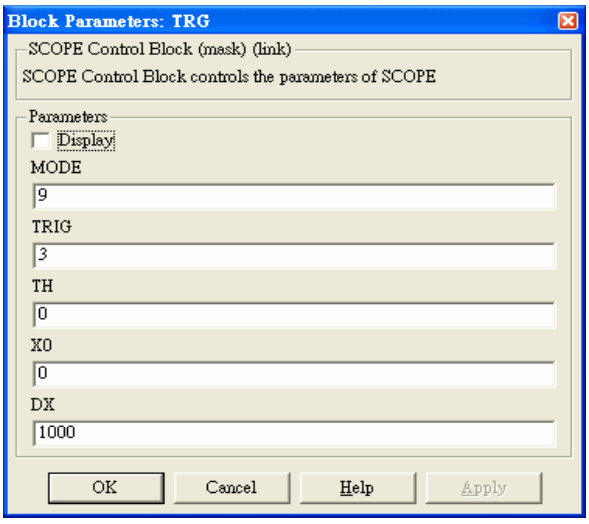


圖 5-2

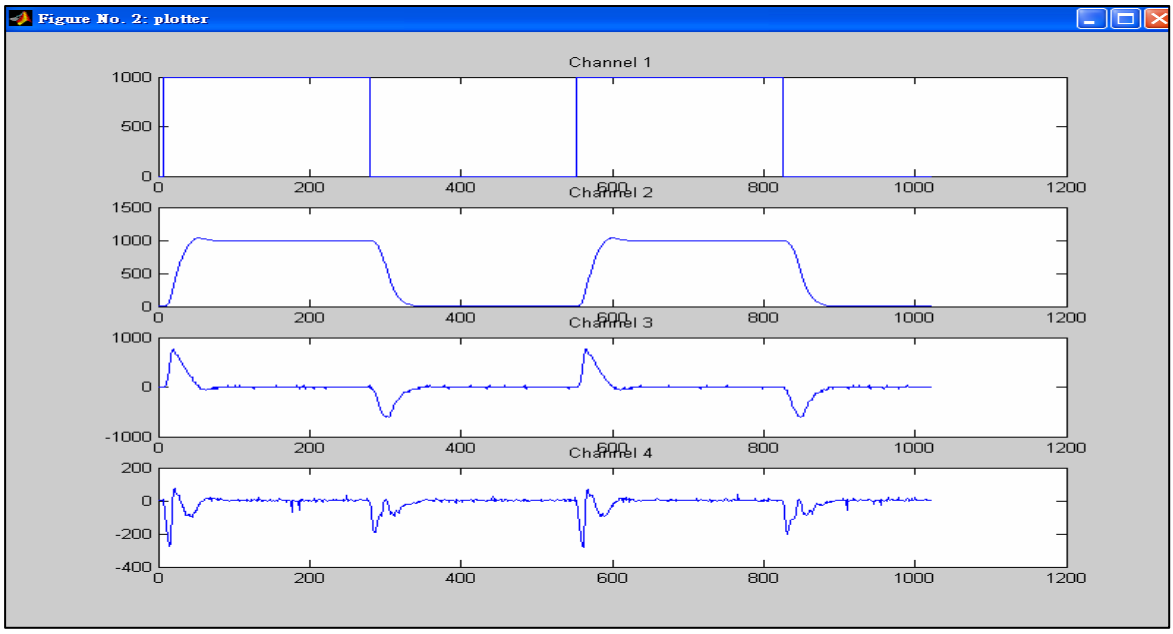
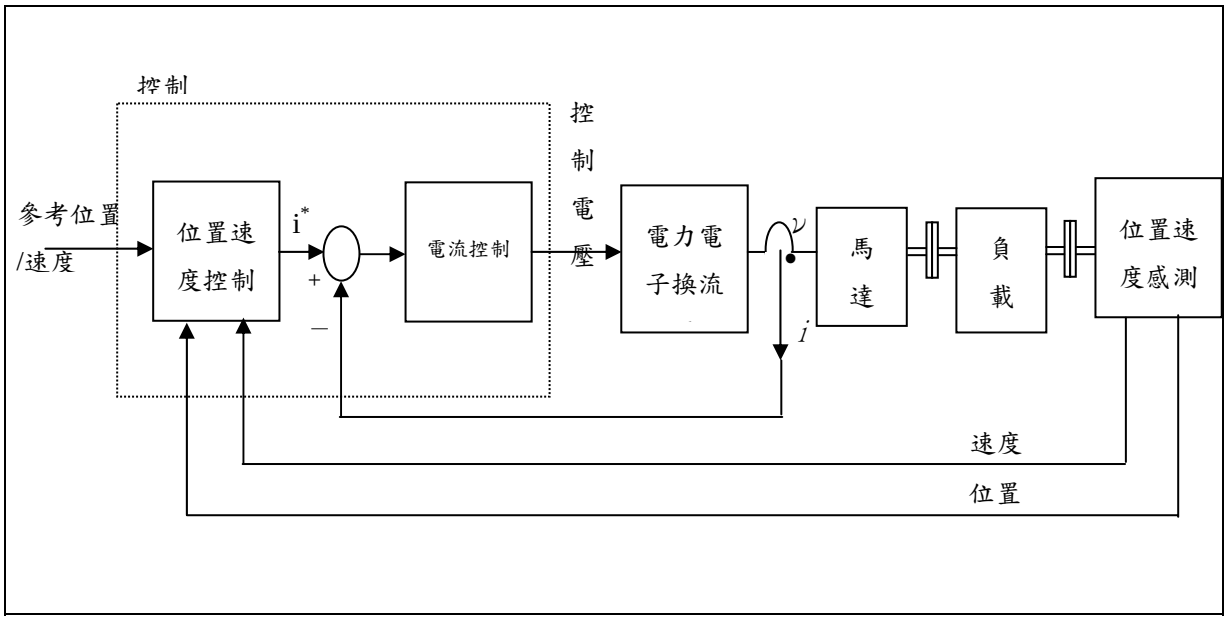


圖 5-3 AC 馬達進行加減速位置控制的實驗數據繪製

9. 實驗說明

圖六所示為交流馬達的控制系統方塊圖，透過電流，速度，以及位置的回授，再搭配適當的控制器可建立完整的閉迴路控制系統。在後面的實驗中將依此一系統方塊圖，逐步建立速度和位置的控制實驗，各實驗的說明和操作方式及參考實驗結果，將逐一進行詳細的說明。



圖三 AC 交流馬達之控制系統方塊圖

#1. 基本參數調整

馬達的基本參數包括：

1.馬達極數(P)：如果電氣角為 θe 而機械角為 θm，則

$\theta e = \theta m * P/2$

所以當 P=4 時，電氣角 2 圈對應為機械角 1 圈。

當 P=8 時，電氣角 4 圈對應為機械角 1 圈。

2.光電盤精度(PPR)：由於雙相解碼的結果，實際上的角度精度是 PPR*4。

所以當 PPR=1000 時，每圈產生 4000-pulse。

當 PPR=2500 時，每圈產生 10000-pulse。

3.電流範圍(A)：因為電流回授的 ADC 精度為 12-bit，只能檢出+/-2048 的數值，所以必須依照電流範圍調整 ADC 增益。

DSP 控制模組中的馬達參數包括：

.io1.zone：PWM 電壓的切換保護(0~63)，單位為 60=5us，預設值=60。

.io2.vz ：V 相電流的 ADC 偏移值，預設值=2048

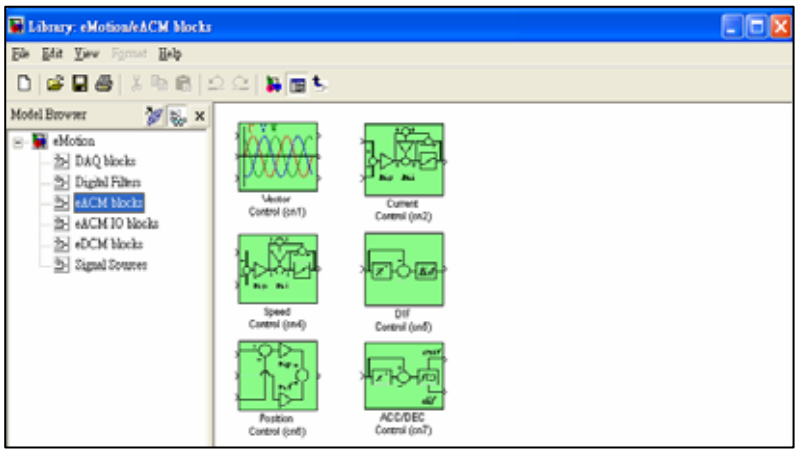
.io2.wz　：W 相電流的 ADC 偏移值，預設值=2048
.io2.vk　：V 相電流的 ADC 增益，單位為 1.0=4096，預設值=4096。
.io2.wk　：W 相電流的 ADC 增益，單位為 1.0=4096，預設值=4096。
.io4.kr　：相位計算增益 Kr=256*P/2/PPR/4*65536，對應表格如下：

馬達極數 P	2	2	4	4	8	8
馬達精度 PPR	1000	2500	1000	2500	1000	2500
相位增益 kr	4194	1678	8388	3355	16777	6711

.cn1.kTq　：測試用的扭力命令，預設值=50。
.cn1.kBias：轉角偏移量，預設值=50。
.cn5.kd　：速度回授的差分增益，單位為 1.0=256，預設值=1024。
.ft1.dt　：開環路速度模擬器的參數，預設值=10。
.ft1.ka　：開環路速度模擬器的參數，預設值=400。
.ft2.dt　：開環路電流模擬器的參數，預設值=1。
.ft2.ka　：開環路電流模擬器的參數，預設值=320。
其他的控制器參數則需再依照下面的測試程序做調整。

另外，在 Matlab\Simulink 提供之圖形介面下，系統所需的控制方塊可由 **c:\smb\eACM8\eMotion.mdl** 中取得。

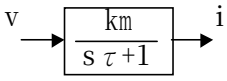
在 Matlab 程式環境中，執行 c:\smb\eACM8\eMotion.mdl 後，可叫出如下圖七所示的模組函式庫，使用者可依需要取出適當的模組(各模組方塊之命名均依該方塊對應之 mod 檔名命名，使用者可依此選擇 block)，並將各模組方塊連線後得出控制系統。



圖四　eMotion 系統在 Simulink 環境下的程式庫

#2. 開迴路電壓控制

電流迴路的開環路特性，可簡化為一階系統處理，方塊圖如下：



其中：

1.其中

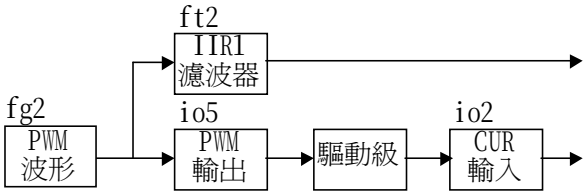
v ：送給驅動器的單相 PWM 電壓命令。

i ：驅動器的單相輸出電流，可由 ADC 偵測。

km：一階系統的穩態增益。

τ：一階系統的時間常數，即步階響應中的上升時間。

為了觀測電流迴路的開環路響應，建立下面的測試方塊圖：



其中：

1.用 fg2 產生測試用的方波訊號。

2.用 ft2 建立對照用的一階系統，即是 IIR1 濾波器。

3.將方波命令產生 PWM 電壓，再觀測驅動級中的電流輸出波形。

4.調整 IIR1 濾波器的參數，使得兩組波形完全相符。這時 IIR1 濾波器就是電流迴路的開環路特性。其中 km 為增益，而 τ 為上升時間(單位為 sec)。

5.增益 km 可從示波器中作簡單計算，而上升時間(τ =4096/4000*dt/ka)的表格列舉如下：

IIR1 參數 dt	10	5	2	1	1	1	1
IIR1 參數 ka	256	256	256	256	512	1024	2048
上升時間 τ	0.04	0.02	0.008	0.004	0.002	0.001	0.0005

6.記錄 km 和 τ ，以進行下一步的計算工作。

對應之 plc 檔程式說明：(exp1_iotest.plc)

```
echo off;                                                          //**畫面顯示開始
run stop;                                                          //**既有程式停止
run testACM;                                                       //**下載指定程式
prog motor.plc;                                                    //**下載馬達參數
echo off;                                                          //**畫面顯示開始
echo clear;
echo "************************************";
echo "*  實驗一: 三相交流電壓的產生        *";
echo "************************************";
echo " ";
echo "模組使用狀況：";
echo "  IO1: ACM_protect.mod (驅動級保護界面)";
echo "  IO2: CUR_input.mod   (三相電流輸入界面)";
echo "  IO3: POS_input.mod   (位置速度輸入界面)";
echo "  IO4: UVW_input.mod   (馬達相位輸入界面)";
echo "  IO5: PWM_output.mod  (PWM 電壓輸出界面)";
echo " ";
echo "  FG3: SAW_wave.mod    (鋸齒波產生器)";
echo "  CN1: VEC_control.mod (三相分配電路)";
echo " ";                                                          //**畫面顯示結束
module .dsp.scope.ch1;                                             //**設定示波器 ch1
s addr &.fg3.out;  s zero 0; s y0 70; s dy 256;
module .dsp.scope.ch2;                                             //**設定示波器 ch2
s addr &.cn1.ucmd; s zero 0; s y0 50; s dy 1000;
module .dsp.scope.ch3;                                             //**設定示波器 ch3
s addr &.cn1.vcmd; s zero 0; s y0 30; s dy 1000;
module .dsp.scope.ch4;                                             //**設定示波器 ch4
s addr &.cn1.wcmd; s zero 0; s y0 10; s dy 1000;
module .dsp.scope;                                                 //**設定示波器觸發和 X 軸
s chn  &.fg3.out; s th  10;  s x0 50;  s dx 11000; s mode 9; s trig -3;

module .io4;                                                       //**設定 io4 模組
s spd  &.io3.spd; s kFrq  150;
module .fg3;                                                       //**設定 fg3 模組
s kAmp 256;
module .cn1;                                                       //**設定 cn1 模組
s ang &.fg3.out; s kBias 0;
module .io5;                                                       //**設定 io5 模組
s vcmd &.cn1.vcmd; s wcmd &.cn1.wcmd;

module .;                                                          //**回復預設模組
run 8khz;                                                          //**設定抽樣頻率為 4KHz
run start;                                                         //**DSP 開始執行
echo on;                                                           //**回復正常訊息顯示
```

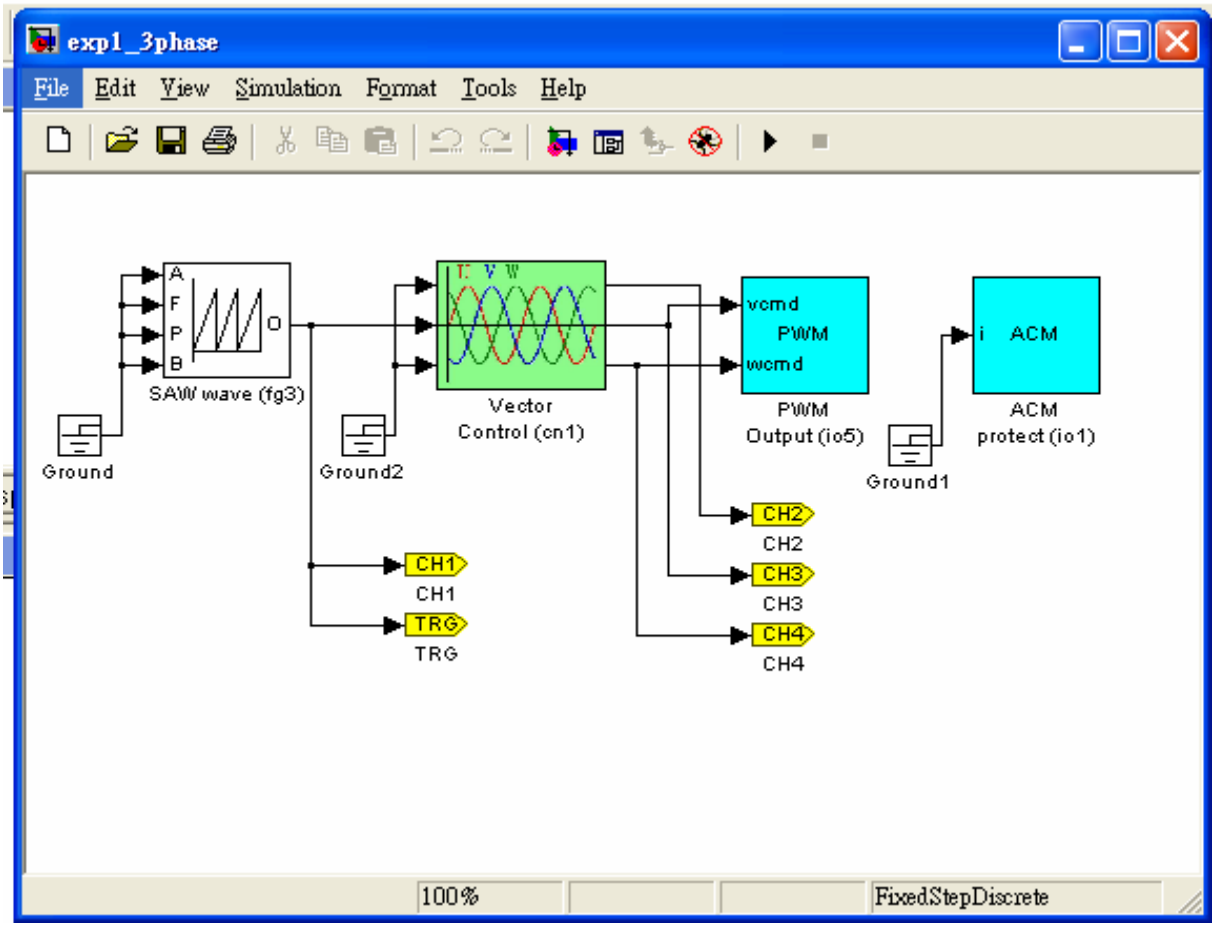
實驗 1.觀測三相交流電壓

對應之 plc 檔程式說明：(exp1_command.plc)

```
echo off;                                                    //**畫面顯示開始
echo clear;
echo "*************************************";
echo "*  ch1: 三角波訊號                    *";
echo "*  ch2: U 相電壓命令                  *";
echo "*  ch3: V 相電壓命令                  *";
echo "*  ch4: W 相電壓命令                  *";
echo "*************************************";
echo "   請注意：PWM 電壓命令恆為正值";
echo " ";                                                    //**畫面顯示結束
module .dsp.scope.ch1;                                       //**設定示波器 ch1
s addr &.fg3.out;  s zero 0; s y0 70; s dy 256;
module .dsp.scope.ch2;                                       //**設定示波器 ch2
s addr 0x53; s zero 0; s y0 50; s dy 1500;
module .dsp.scope.ch3;                                       //**設定示波器 ch3
s addr 0x54; s zero 0; s y0 30; s dy 1500;
module .dsp.scope.ch4;                                       //**設定示波器 ch4
s addr 0x55; s zero 0; s y0 10; s dy 1500;
module .dsp.scope;                                           //**設定示波器觸發和 X 軸
s chn  &.fg3.out; s th  10;  s x0 50;  s dx 11000; s mode 9; s trig -3;
module .;                                                    //**回復預設模組
echo on;                                                     //**恢復正常訊息顯示
```

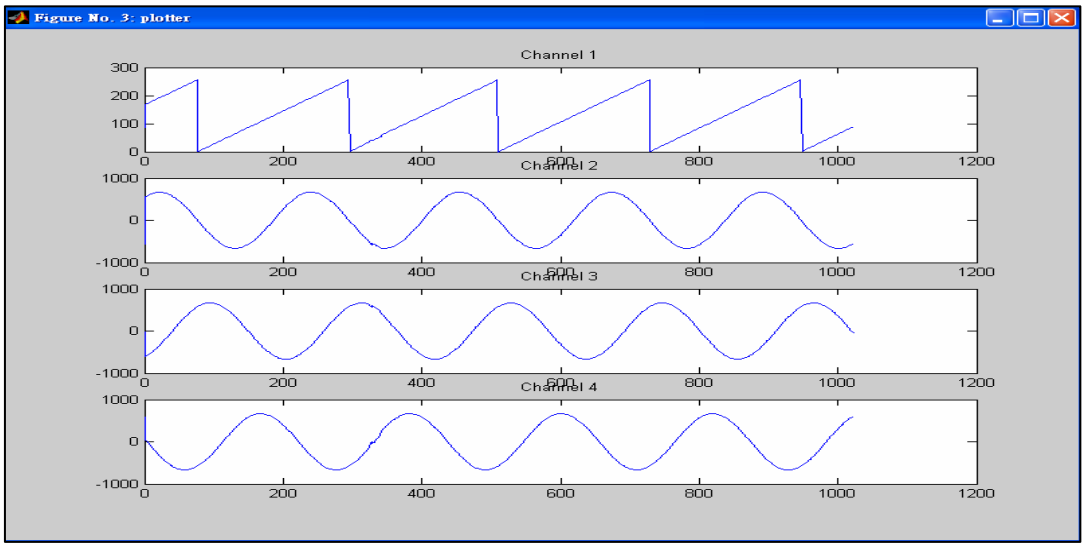
對應之 Simulink mdl 檔說明：(exp1_3phase.mdl)

依照實驗說明中的功能方塊圖，到 c:\smb\eACM8\eMotion.mdl 中選取對應的方塊圖，建立如下圖中的控制方塊圖，此控制系統的功能和 plc 檔中所描述的系統功能相同。但在圖形介面的輔助下，使用者可用更直覺更快速的方式建立功能模組間的連線關係，而且變更模組參數時也較方便。



圖五 實驗 1 – 開迴路的電壓控制三相電壓輸出

實測響應圖：



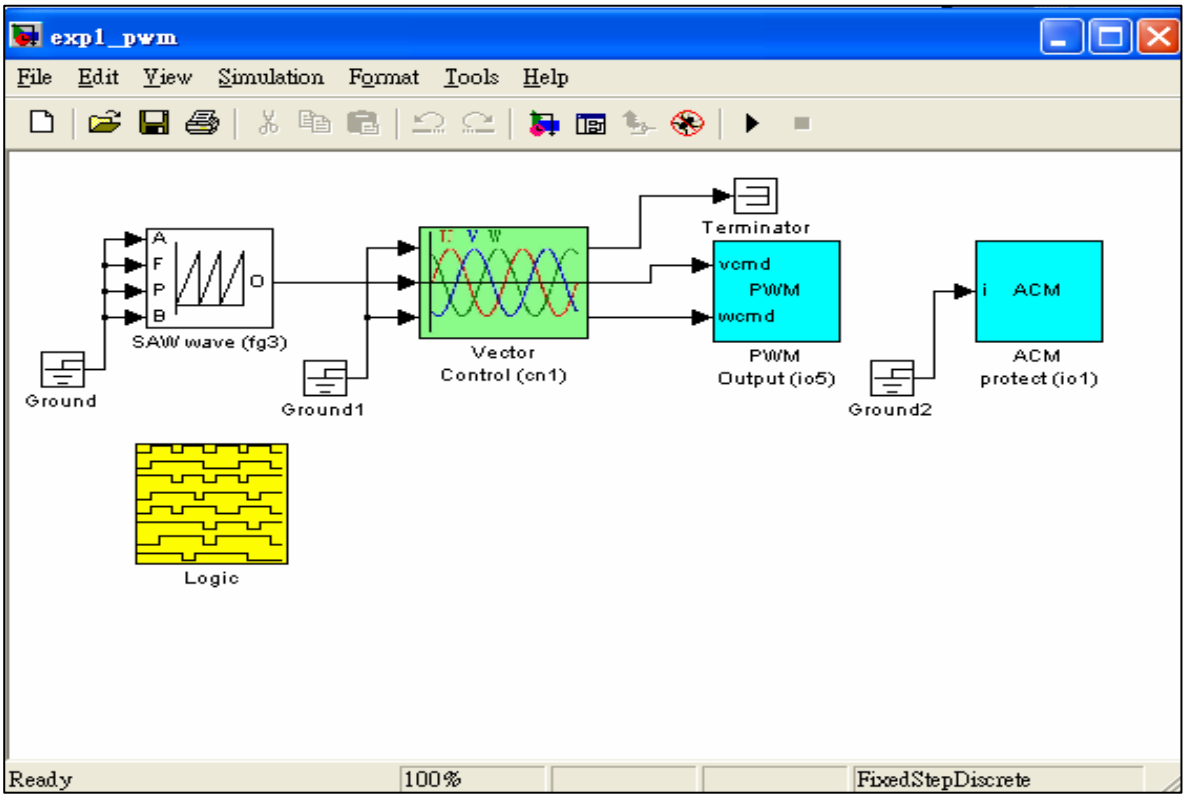
圖六 交流馬達之開迴路電壓控制三相電壓響應圖

實驗 2.觀測三相 PWM 波形

對應之 plc 檔程式說明：(exp1_pwm.plc)

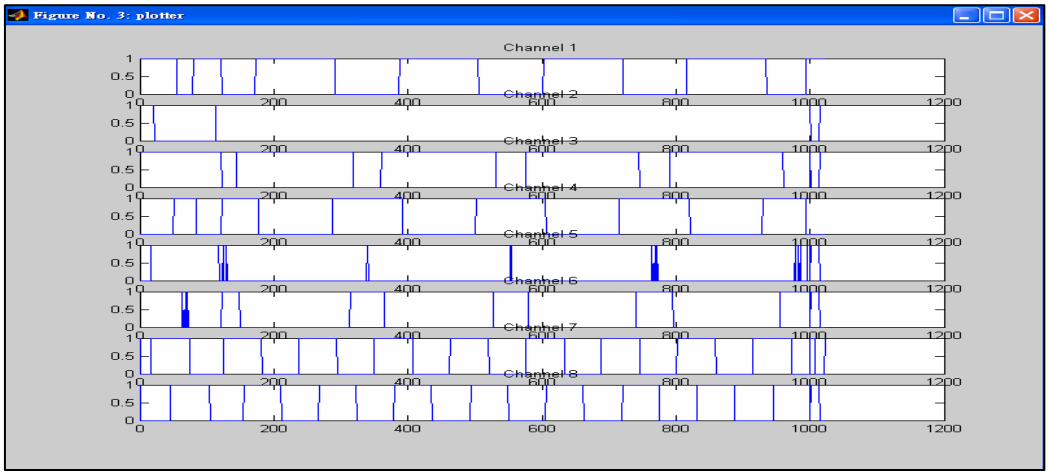
```
echo off;                                                    //**畫面顯示開始
echo clear;
echo "*************************************";
echo "*  ch0: PWM+U 訊號                    *";
echo "*  ch1: PWM+V 訊號                    *";
echo "*  ch2: PWM+W 訊號                    *";
echo "*  ch3: PWM-U 訊號                    *";
echo "*  ch4: PWM-V 訊號                    *";
echo "*  ch5: PWM-W 訊號                    *";
echo "*  ch6: 光電盤 A 相訊號                *";
echo "*  ch7: 光電盤 B 相訊號                *";
echo "*************************************";
echo " ";
echo " ";                                                    //**畫面顯示結束
module .dsp.scope;                                           //**設定示波器觸發和 X 軸
s chn  0; s th 0;  s x0 50;  s dx 50; s mode 2; s trig 3;
echo on;                                                     //**恢復正常訊息顯示
```

對應之 Simulink mdl 檔說明：(exp1_PWM.mdl)



圖十一實驗 1.開迴路的電壓控制 PWM 輸出

實測響應圖：



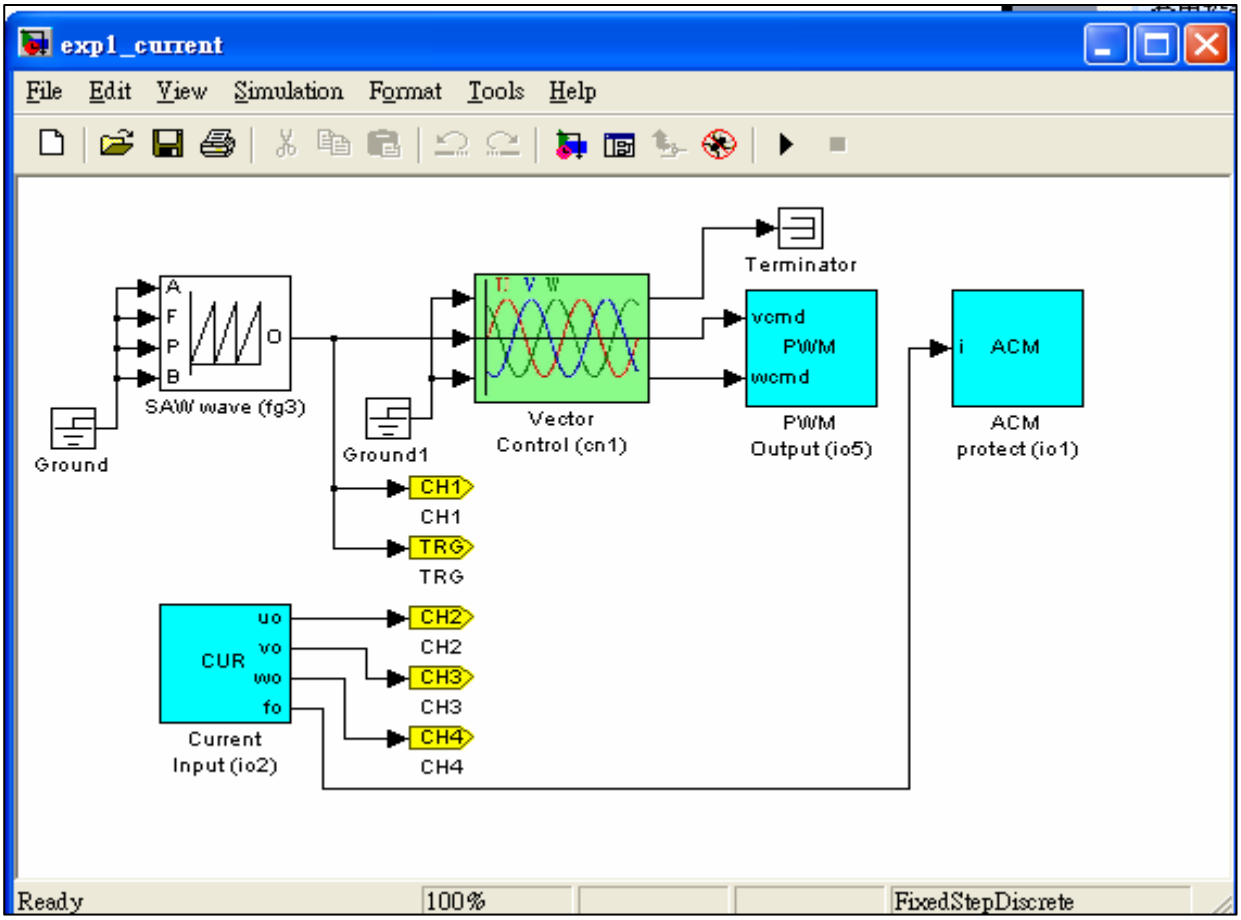
圖十一－開迴路的電壓控制 PWM 輸出響應圖

實驗 3.觀測三相電流迴授

對應之 plc 檔程式說明：(exp1_current.plc)

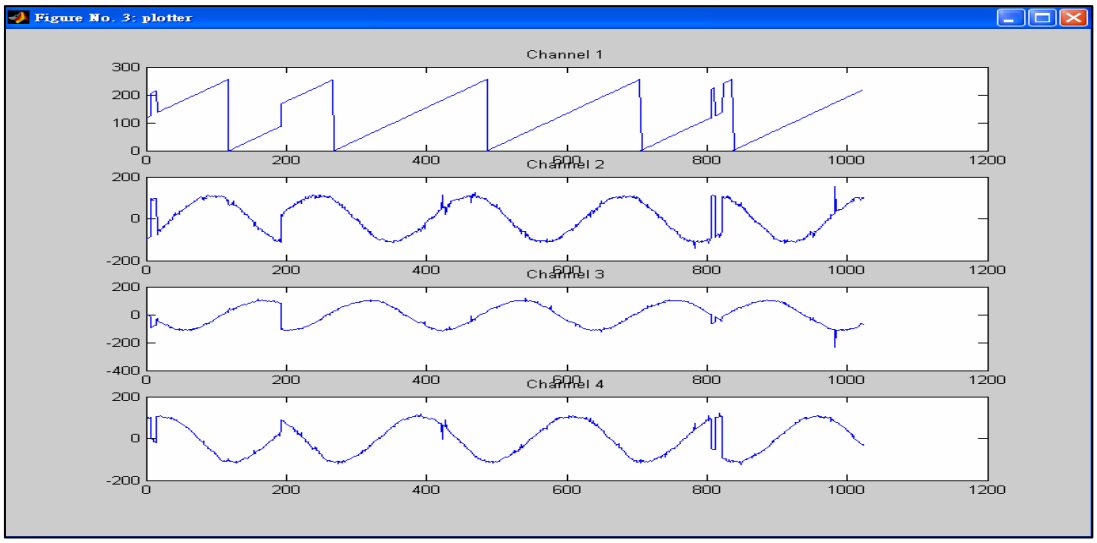
```
echo off;                                                    //**畫面顯示開始
echo clear;
echo "*************************************";
echo "*  ch1: 三角波訊號                    *";
echo "*  ch2: U 相電流回授                  *";
echo "*  ch3: V 相電流回授                  *";
echo "*  ch4: W 相電流回授                  *";
echo "*************************************";
echo " ";
echo " ";                                                    //**畫面顯示結束
module .dsp.scope.ch1;                                       //**設定示波器 ch1
s addr &.fg3.out; s zero 0; s y0 70; s dy 256;
module .dsp.scope.ch2;                                       //**設定示波器 ch2
s addr &.io2.uo;  s zero 0; s y0 50; s dy 100;
module .dsp.scope.ch3;                                       //**設定示波器 ch3
s addr &.io2.vo;  s zero 0; s y0 30; s dy 100;
module .dsp.scope.ch4;                                       //**設定示波器 ch4
s addr &.io2.wo;  s zero 0; s y0 10; s dy 100;
module .dsp.scope;                                           //**設定示波器觸發和 X 軸
s chn  &.fg3.out; s th  10;  s x0 50;  s dx 11000; s mode 9; s trig -3;
echo on;                                                     //**恢復正常訊息顯示
```

對應之 Simulink mdl 檔說明：(exp1_current.mdl)



圖十二－實驗 1. 開迴路的電壓控制三相電流量測

實測響應圖：



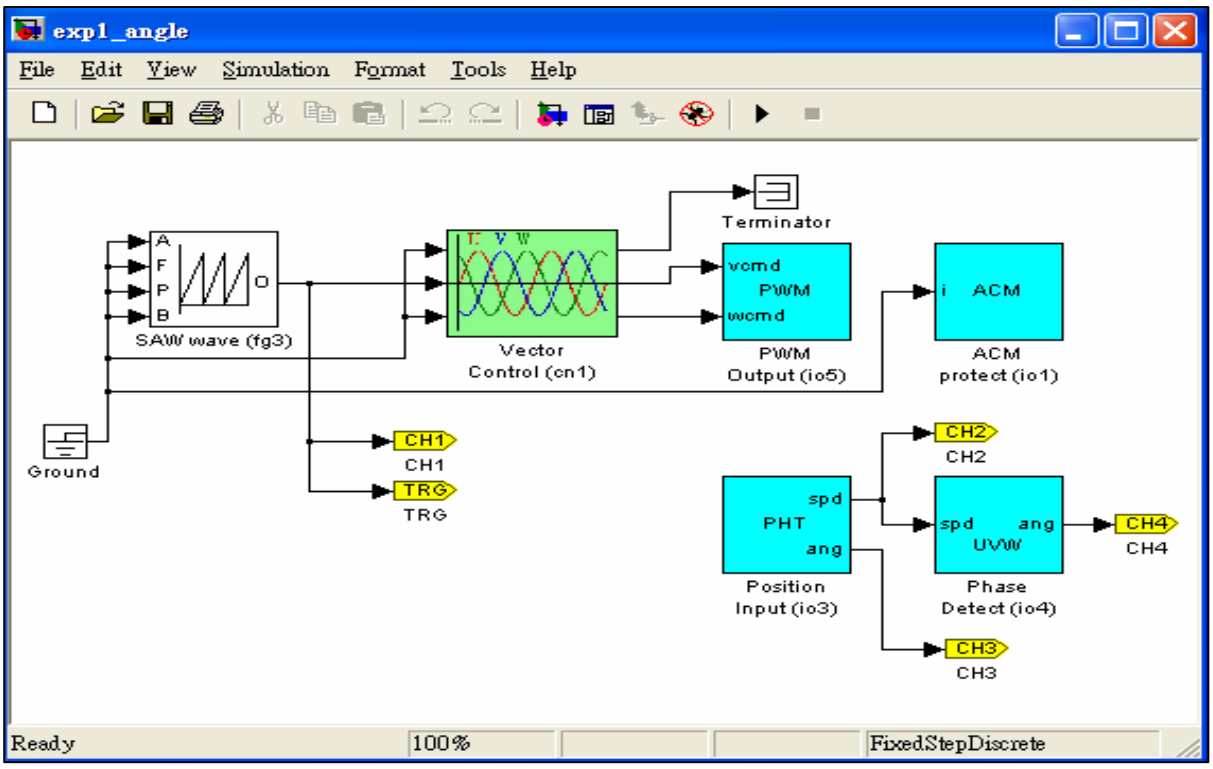
圖十三－開迴路的電壓控制三相電流量測響應圖

實驗 1.觀測速度位置轉角

對應之 plc 檔程式說明：(exp1_angle.plc)

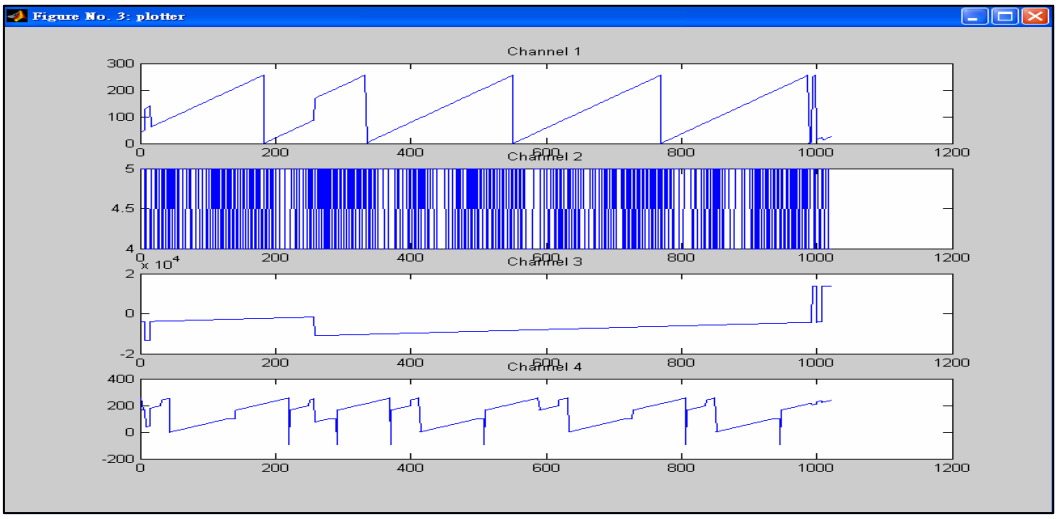
```
echo off;                                                    //**畫面顯示開始
echo clear;
echo "************************************";
echo "*  ch1: 三角波                      *";
echo "*  ch2: 馬達速度回授                *";
echo "*  ch3: 馬達位置回授                *";
echo "*  ch4: 馬達轉角回授                *";
echo "************************************";
echo " ";
echo " ";                                                    //**畫面顯示結束
module .dsp.scope.ch1;                                       //**設定示波器 ch1
s addr &.fg3.out; s zero 0; s y0 70; s dy 256;
module .dsp.scope.ch2;                                       //**設定示波器 ch2
s addr &.io3.spd; s zero 0; s y0 50; s dy 10;
module .dsp.scope.ch3;                                       //**設定示波器 ch3
s addr &.io3.ang; s zero 0; s y0 30; s dy 30000;
module .dsp.scope.ch4;                                       //**設定示波器 ch4
s addr &.io4.ang; s zero 0; s y0 10; s dy 250;
module .dsp.scope;                                           //**設定示波器觸發和 X 軸
s chn  &.fg3.out; s th  10;  s x0 50;  s dx 11000; s mode 9; s trig -3;
echo on;                                                     //**恢復正常訊息顯示
```

對應之 Simulink mdl 檔說明：(exp1_angle.mdl)



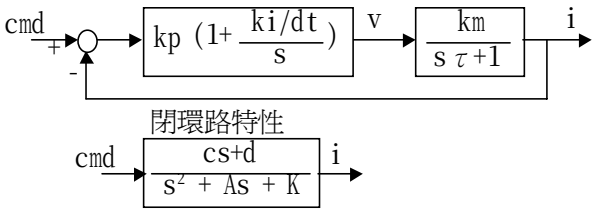
圖十四－ 實驗 1. 開迴路的電壓控制速度位置轉角

實測響應圖：



圖十五－ 開迴路的電壓控制速度位置轉角響應圖

#3.開環路的定位控制



電流迴路搭配 PI 控制器作回授處理，方塊圖如下：

其中：

1.PI 控制器中有 kp 和 ki 兩項參數需要設計。

2.在閉環路的轉移函數中，分子項不需處理，而分母項的處理原則如下：

A = 2*0.7*wn

K = wn*wn

其中 wn 是兩階系統的頻寬，一般而言(wn=500)是最常用的設計需求，這時 A=700 而 K=250000。

3.對照開環路和閉環路的參數，取得

A = (1+kp*km) / τ

K = kp * km * ki / τ / dt

4.所以：

kp = (A* τ -1)/km

ki = K* τ *dt/kp/km = K* τ *dt/(A* τ -1)

可見 ki 值只和開環路的上升時間 τ 有關，和增益 km 無關。

5.當 wn=500、dt=1/4000、 τ =0.0032、km=0.6 時，計算出 kp=2.1 而 ki=0.16。

6.因為

kp 的單位為 1.0=4096，而

ki 的單位為 1.0=65536，

所以實際上 kp=8602 而 ki=10486。

對於輸出的上下限，因為 PI 控制器的輸出為 PWM 電壓命令，而 PWM 產生器的命令範圍 0~3072，所以將 PI 控制器的上下限固定設為

high：輸出上限，設為 1500。

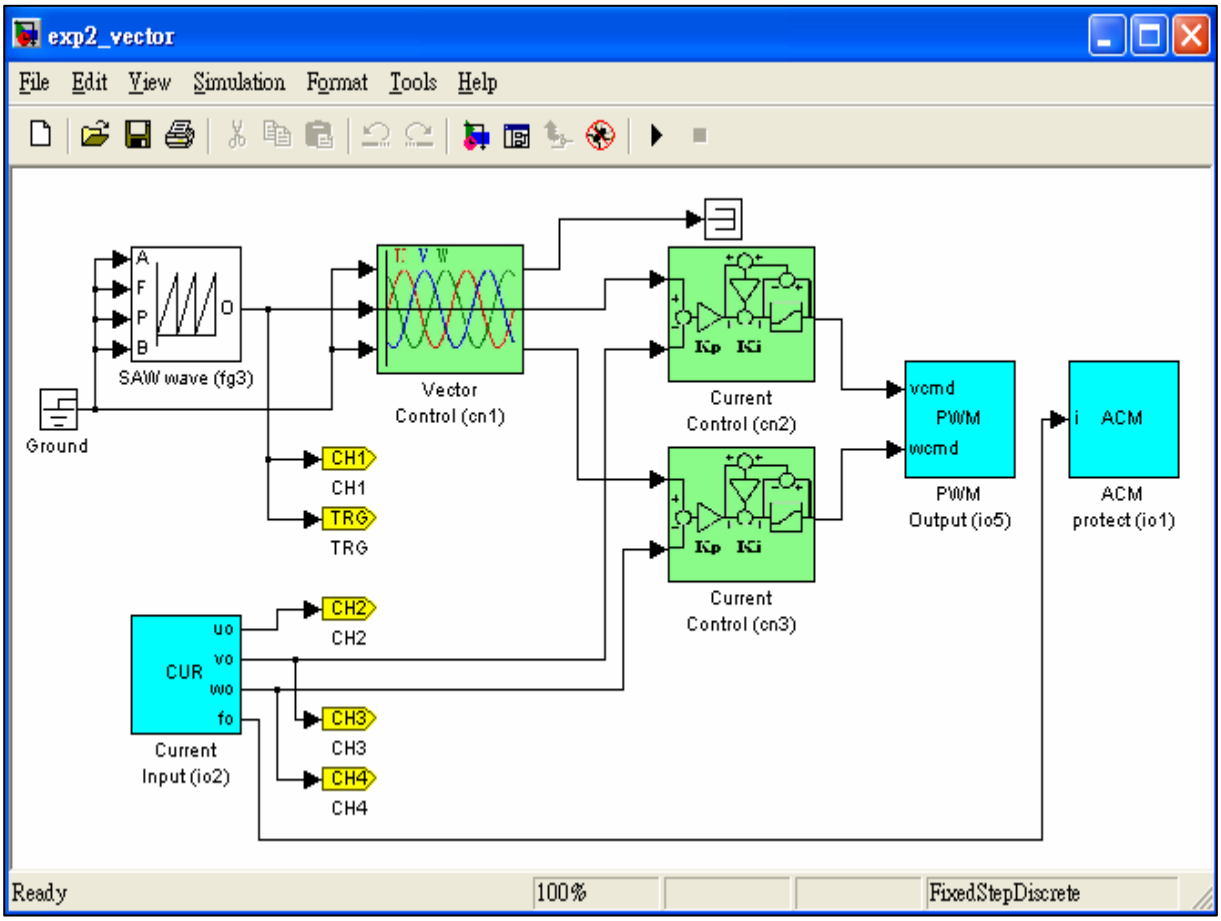
low ：輸出下限，設為-1500。

對應之 plc 檔程式說明：(exp2_vertor.plc)

```
echo off;                                                    //**畫面顯示開始
run stop;                                                    //**既有程式停止
run testACM;                                                 //**下載指定程式
prog motor.plc;                                              //**下載馬達參數
echo clear;                                                  //**畫面清除
echo "*****************************************";
echo "*  實驗二: 開環路的定位控制(步進馬達模式) *";
echo "*****************************************";
echo " ";
echo "模組使用狀況：";
echo " ";
echo "  FG3: SAW_wave.mod    (鋸齒波產生器)";
echo "  CN1: VEC_control.mod (三相分配電路)";
echo "  CN2: CUR_control.mod (V 相電流控制)";
echo "  CN3: CUR_control.mod (W 相電流控制)";
echo " ";
echo " 示波器：ch2=U 相電流，ch3=V 相電流，ch4=W 相電流";
echo " ";                                                    //**畫面顯示結束

module .dsp.scope.ch1;                                       //**設定示波器 ch1
s addr &.fg3.out;  s zero 0; s y0 70; s dy 256;
module .dsp.scope.ch2;                                       //**設定示波器 ch2
s addr &.io2.uo;   s zero 0; s y0 50; s dy 150;
module .dsp.scope.ch3;                                       //**設定示波器 ch3
s addr &.io2.vo;   s zero 0; s y0 30; s dy 150;
module .dsp.scope.ch4;                                       //**設定示波器 ch4
s addr &.io2.wo;   s zero 0; s y0 10; s dy 150;
module .dsp.scope;                                           //**設定示波器觸發和 X 軸
s chn  &.fg3.out; s th  10;  s x0 50;  s dx 12000; s mode 9; s trig -3;

module .io4;                                                 //**設定 io4 模組
s spd  &.io3.spd;
module .fg3;                                                 //**設定 fg3 模組
s kAmp 256; s kFrq 50;                          //**10 Hz
module .cn1;                                                 //**設定 cn1 模組
s ang &.fg3.out; s kTq 150; s kBias 0;
module .cn2;                                                 //**設定 cn2 模組
s cmd &.cn1.vcmd; s fdk &.io2.vo;
module .cn3;                                                 //**設定 cn3 模組
```

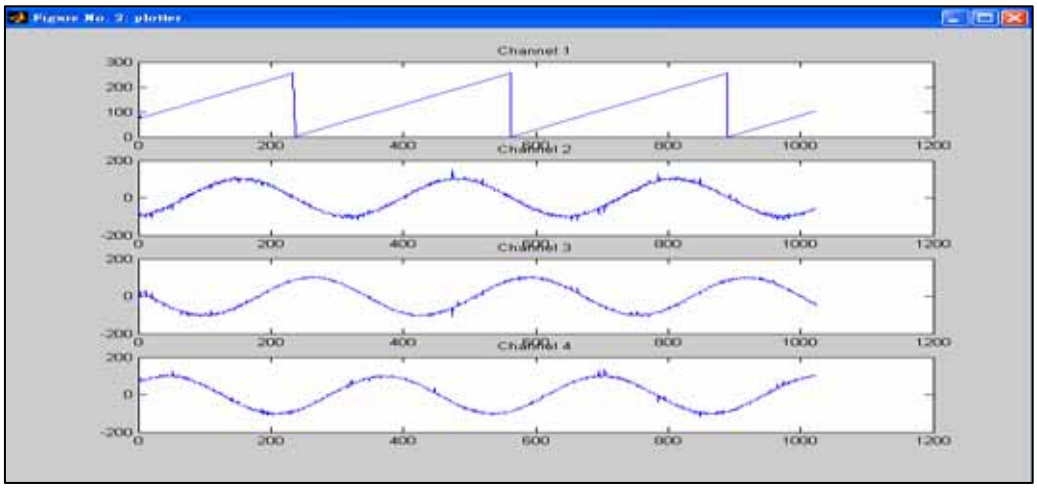
```
s cmd &.cn1.wcmd; s fdk &.io2.wo;
module .io5;                                          //**設定 io5 模組
s vcmd &.cn2.out; s wcmd &.cn3.out;

module .;                                             //**回復預設模組
run 8khz;                                             //**設定抽樣頻率為 4KHz
run start;                                            //**DSP 開始執行
echo on;                                              //**回復正常訊息顯示
```

對應之 Simulink mdl 檔說明：(exp2_vertor.mdl)



圖十六－ 實驗 2. 開迴路的定位控制

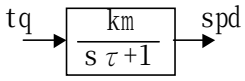
實測響應圖：



圖十七－ 開迴路的定位控制響應圖

#4. 閉環路的扭力控制

速度迴路的開環路特性雖然是高階系統，但是還是簡化為一階系統做處理，方塊圖如下：



其中：

1.其中

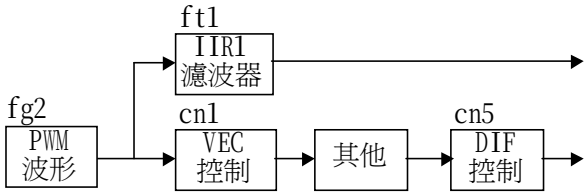
tq ：送給相位分配的扭力命令。

spd：馬達的速度回授，可由角度作差分而得。

km：一階系統的穩態增益。

τ：一階系統的時間常數，即步階響應中的上升時間。

為了觀測速度迴路的開環路響應，建立下面的測試方塊圖：



其中：

1.用 fg2 產生測試用的方波訊號。

2.用 ft1 建立對照用的一階系統，即是 IIR1 濾波器。

3.用方波產生扭力命令控制三相電流後，再觀測馬達的速度回授波形。

4.調整 IIR1 濾波器的參數，使得兩組波形完全相符。這時 IIR1 濾波器就是速度迴路的開環路特性。其中 km 為增益，而 τ 為上升時間(單位為 sec)。

5.增益 km 可從示波器中作簡單計算，而上升時間(τ =4096/4000*dt/ka)的表格列舉如下：

IIR1 參數 dt	10	10	10	10	10
IIR1 參數 ka	125	250	500	1000	2000
上升時間 τ	0.08	0.04	0.02	0.01	0.005

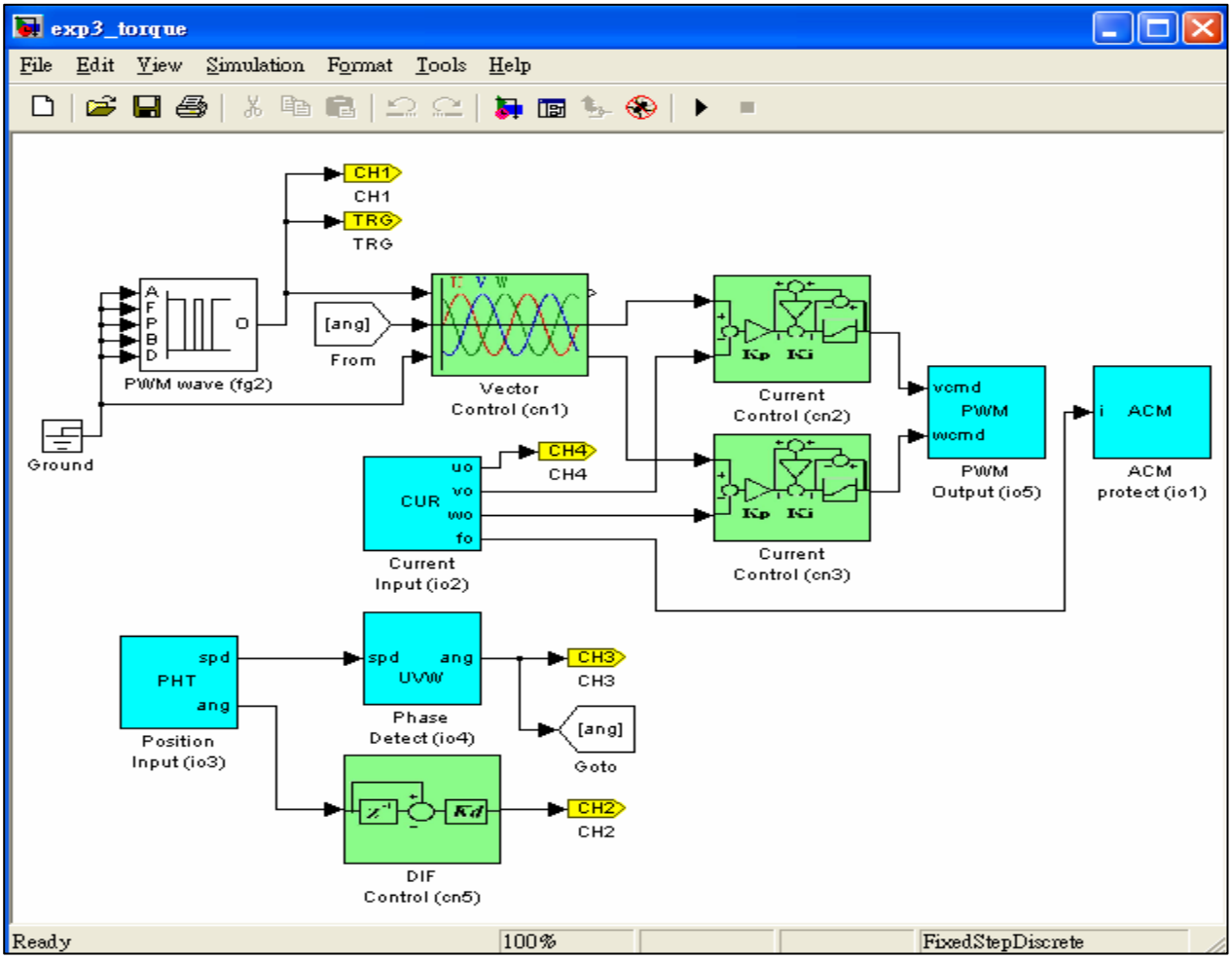
6.記錄 km 和 τ，以進行下一步的計算工作。

對應之 plc 檔程式說明：(exp3_torque.plc)

```
echo off;                                                          //**畫面顯示開始
run stop;                                                          //**既有程式停止
run testACM;                                                       //**下載指定程式
prog motor.plc;                                                    //**下載馬達參數
echo clear;                                                        //**畫面清除
echo "**********************************";
echo "*  實驗三: 閉環路的扭力控制        *";
echo "**********************************";
echo " ";
echo "模組使用狀況：";
echo " ";
echo "  FG2: PWM_wave.mod    (方波產生器)";
echo "  CN1: VEC_control.mod (三相分配電路)";
echo "  CN2: CUR_control.mod (V 相電流控制)";
echo "  CN3: CUR_control.mod (W 相電流控制)";
echo "  CN5: DIF_control.mod (差分控制器)";
echo " ";
echo " 示波器：ch1=扭力命令，ch2=速度回授，ch3=電流回授，ch4=馬達轉角";
echo " ";                                                          //**畫面顯示結束

module .dsp.scope.ch1;                                             //**設定示波器 ch1
s addr &.fg2.out;  s zero 0; s y0 70; s dy 100;
module .dsp.scope.ch2;                                             //**設定示波器 ch2
s addr &.cn5.dif;  s zero 0; s y0 50; s dy 2000;
module .dsp.scope.ch3;                                             //**設定示波器 ch3
s addr &.io4.ang;  s zero 0; s y0 30; s dy 256;
module .dsp.scope.ch4;                                             //**設定示波器 ch4
s addr &.io2.uo;   s zero 0; s y0 10; s dy 100;
module .dsp.scope;                                                 //**設定示波器觸發和 X 軸
s chn  &.fg2.out; s th 10;  s x0 0;  s dx 15000; s mode 9; s trig 2;
module .io4;                                                       //**設定 io4 模組
s spd  &.io3.spd;

module .fg2;                                                       //**設定 fg2 模組
s kAmp 100; s kFrq 8; s kDuty 64;
module .cn1;                                                       //**設定 cn1 模組
s ang &.io4.ang; s tq &.fg2.out; s kBias 0;
module .cn2;                                                       //**設定 cn2 模組
s cmd &.cn1.vcmd; s fdk &.io2.vo;
module .cn3;                                                       //**設定 cn3 模組
```

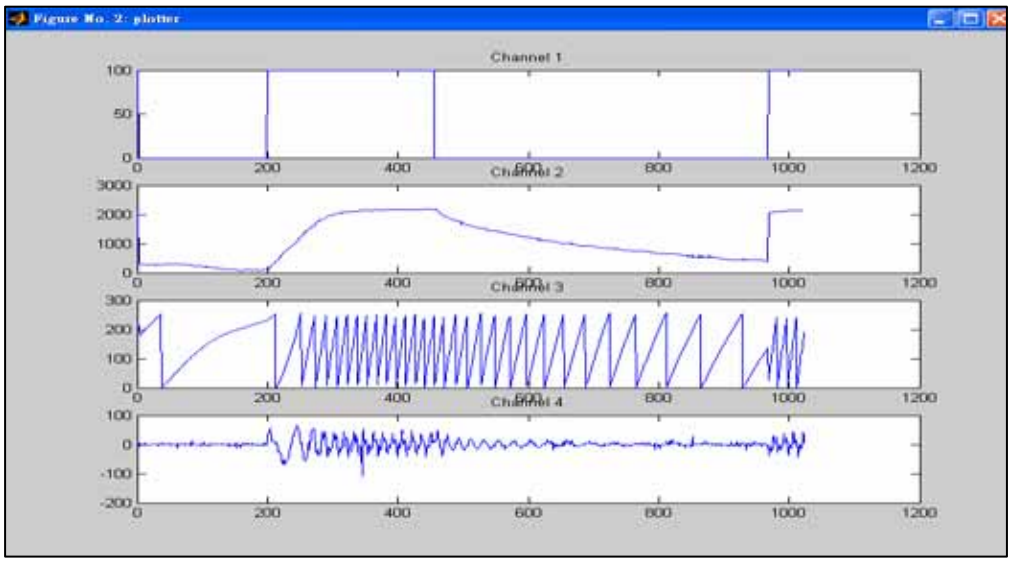
```
s cmd &.cn1.wcmd; s fdk &.io2.wo;
module .io5;                                        //**設定 io5 模組
s vcmd &.cn2.out; s wcmd &.cn3.out;
module .ft1;                                        //**設定 ft1 模組
s u   &.fg2.out;
module  .cn5;                                       //**設定 cn5 模組
s inp &.io3.ang;

module .;                                           //**回復預設模組
run 8khz;                                           //**設定抽樣頻率為 4KHz
run start;                                          //**DSP 開始執行
echo on;                                            //**回復正常訊息顯示
```

對應之 Simulink mdl 檔說明：(exp3_torque.mdl)



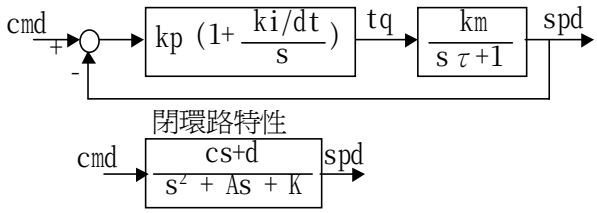
圖十八－ 實驗 3 .閉環路的扭力控制

實測響應圖：



圖十一七－閉環路的扭力控制響應圖

#5. 速度迴路的閉環路設計



速度迴路搭配 PI 控制器作回授處理，方塊圖如下：

其中：

1.PI 控制器中有 kp 和 ki 兩項參數需要設計。

2.在閉環路的轉移函數中，分子項不需處理，而分母項的處理原則如下：

A = 2*0.7*wn

K = wn*wn

其中 wn 是兩階系統的頻寬，一般而言(wn=120)是最常用的設計需求，這時 A=168 而 K=14400。

3.對照開環路和閉環路的參數，取得

A = (1+kp*km) / τ

K = kp * km * ki / τ / dt

4.所以：

kp = (A*τ-1)/km

ki = K*τ*dt/kp/km = K*τ*dt/(A*τ-1)

可見 ki 值只和開環路的上升時間τ有關，和增益 km 無關。

5.當 wn=120、dt=1/1000、τ=0.03、km=6 時，計算出 kp=0.67 而 ki=0.107。

6.因為

kp 的單位為 1.0=4096，而

ki 的單位為 1.0=65536，

所以實際上 kp=2744 而 ki=7008。

對於輸出的上下限，因為 PI 控制器的輸出為扭力命令，再轉換成三相的電流命令。而電流回授值的範圍為+/-2048，所以將 PI 控制器的上下限固定設為

high：輸出上限，設為 2000。

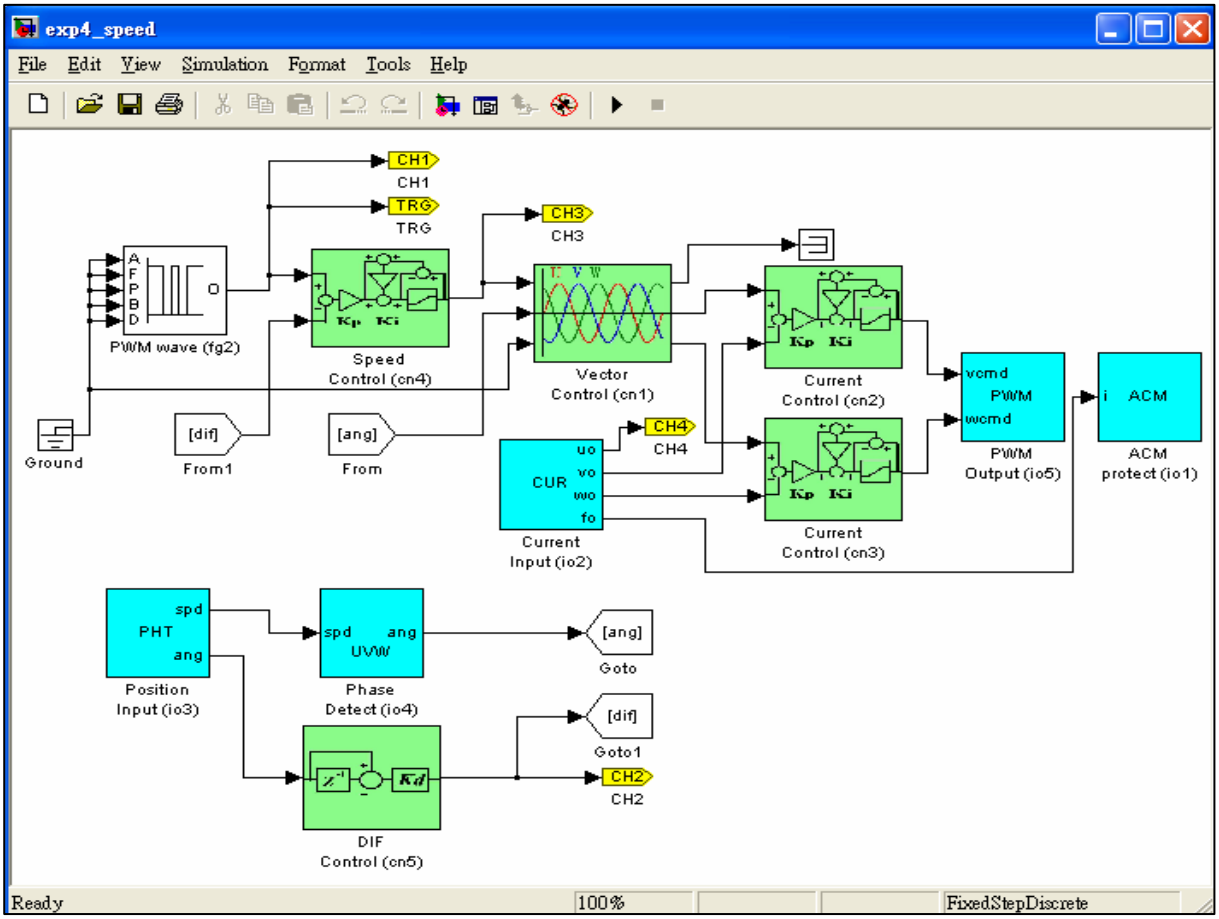
low ：輸出下限，設為-2000。

對應之 plc 檔程式說明：(exp4_speed.plc)

```
echo off;                                                    //**畫面顯示開始
run stop;                                                    //**既有程式停止
run testACM;                                                 //**下載指定程式
prog motor.plc;                                              //**下載馬達參數
echo clear;                                                  //**畫面清除
echo "**********************************";
echo "*  實驗四: 閉環路的速度控制        *";
echo "**********************************";
echo " ";
echo "模組使用狀況：";
echo " ";
echo "  FG2: PWM_wave.mod    (方波產生器)";
echo "  CN1: VEC_control.mod (三相分配電路)";
echo "  CN2: CUR_control.mod (V 相電流控制)";
echo "  CN3: CUR_control.mod (W 相電流控制)";
echo "  CN4: SPD_control.mod (速度控制器)";
echo "  CN5: DIF_control.mod (差分控制器)";
echo " ";
echo " 示波器：ch2=速度回授，ch3=扭力命令，ch4=電流回授";
echo " ";                                                    //**畫面顯示結束

module .dsp.scope.ch1;                                       //**設定示波器 ch1
s addr &.fg2.out;  s zero 0; s y0 70; s dy 1000;
module .dsp.scope.ch2;                                       //**設定示波器 ch2
s addr &.cn5.dif;  s zero 0; s y0 50; s dy 1000;
module .dsp.scope.ch3;                                       //**設定示波器 ch3
s addr &.cn4.out;  s zero 0; s y0 30; s dy 1000;
module .dsp.scope.ch4;                                       //**設定示波器 ch4
s addr &.io2.uo;   s zero 0; s y0 10; s dy 500;
module .dsp.scope;                                           //**設定示波器觸發和 X 軸
s chn  &.fg2.out;  s th 100; s x0 10;  s dx 15000; s mode 9; s trig 2;

module .io4;                                                 //**設定 io4 模組
s spd  &.io3.spd;
module .fg2;                                                 //**設定 fg2 模組
```

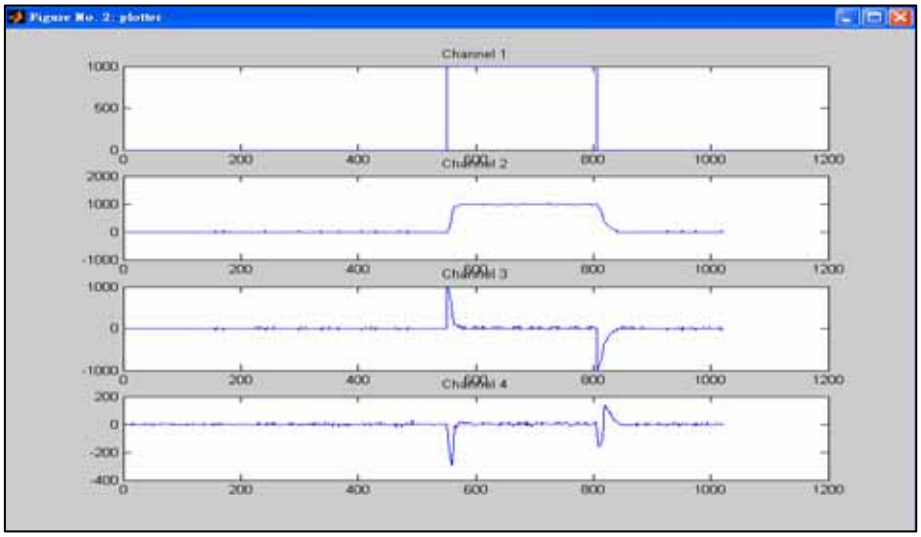
```
s kAmp 1000; s kFrq 8; s kDuty 64;
module .cn1;                                          //**設定 cn1 模組
s ang &.io4.ang; s tq &.cn4.out; s kBias 0;
module .cn2;                                          //**設定 cn2 模組
s cmd &.cn1.vcmd; s fdk &.io2.vo;
module .cn3;                                          //**設定 cn3 模組
s cmd &.cn1.wcmd; s fdk &.io2.wo;
module  .cn4;                                         //**設定 cn4 模組
s cmd &.fg2.out;  s fdk &.cn5.dif;
module  .cn5;                                         //**設定 cn5 模組
s inp &.io3.ang;
module .io5;                                          //**設定 io5 模組
s vcmd &.cn2.out; s wcmd &.cn3.out;
module .ft1;                                          //**設定 ft1 模組
s u   &.fg2.out; s ka 1280;

module .;                                             //**回復預設模組
run 8khz;                                             //**設定抽樣頻率為 4KHz
run start;                                            //**DSP 開始執行
echo on;                                              //**回復正常訊息顯示
```

對應之 Simulink mdl 檔說明：(exp4_speed.mdl)



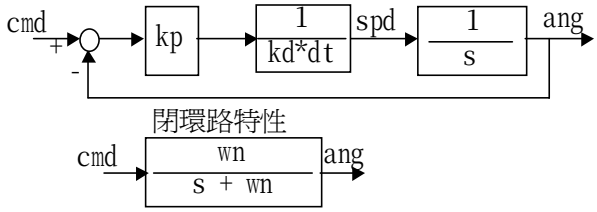
圖二十－實驗 4. 閉環路的速度控制

實測響應圖：



圖二十一－閉環路的速度控制響應圖

#6. 位置迴路的閉環路設計



位置迴路搭配 P 控制器作回授處理，方塊圖如下：

其中：

1.P 控制器中有 kp 一項參數需要設計。

2.在閉環路的轉移函數中，wn 是一階系統的頻寬，一般而言(wn=30)是最常用的設計需求。

3.對照開環路和閉環路的參數，取得

wn = kp/kd/dt

其中抽樣週期固定為 dt=1/1000，而 kd 是速度差分計算上的回授增益。

4.所以：

kp = wn*kd*dt

5.當 wn=30、dt=1/1000、kd=4 時，計算出 kp=0.12。

差分增益 kd	1	2	4	8	16	32
響應頻寬 wn	31	31	31	31	31	31
控制增益 kp	0.03	0.06	0.12	0.25	0.5	1

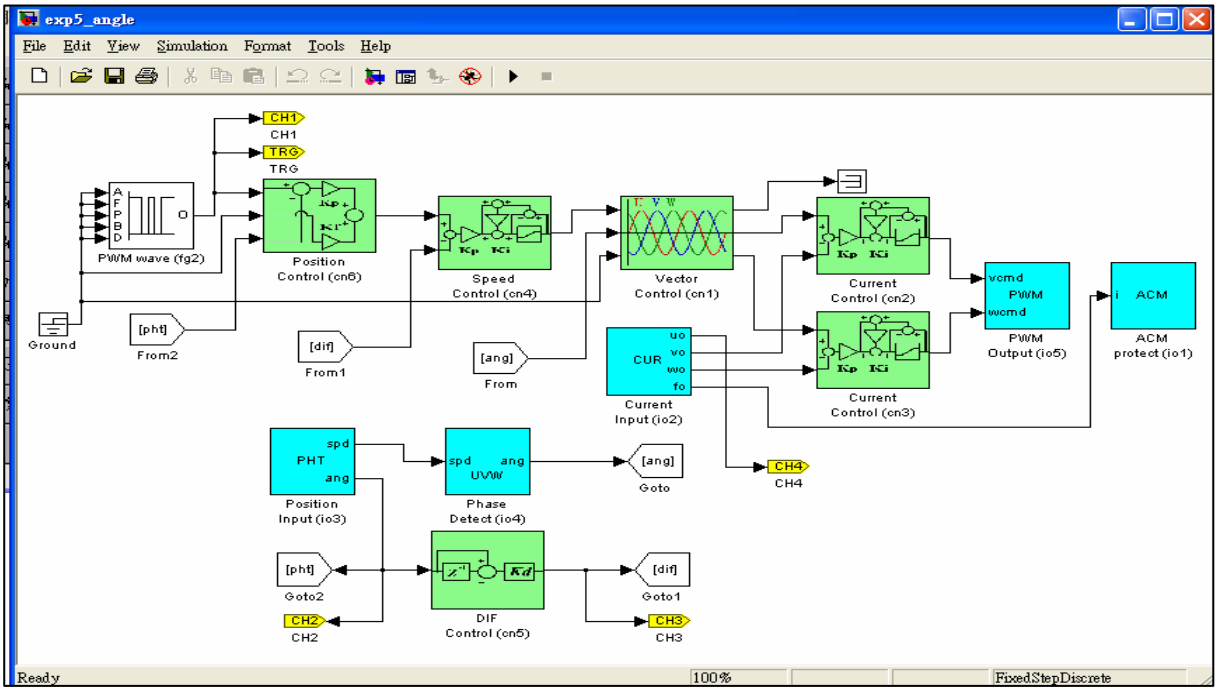
6.因為 kp 的單位為 1.0=4096，所以實際上 kp=512。

對應之 plc 檔程式說明：(exp5_angle.plc)

```
echo off;                                                    //**畫面顯示開始
run stop;                                                    //**既有程式停止
run testACM;                                                 //**下載指定程式
prog motor.plc;                                              //**下載馬達參數
echo clear;                                                  //**畫面清除
echo "**********************************";
echo "*  實驗五: 閉環路的位置控制       *";
echo "**********************************";
echo " ";
echo "模組使用狀況：";
echo " ";
echo "  FG2: PWM_wave.mod    (正弦波產生器)";
echo "  CN1: VEC_control.mod (三相分配電路)";
echo "  CN2: CUR_control.mod (V 相電流控制)";
echo "  CN3: CUR_control.mod (W 相電流控制)";
echo "  CN4: SPD_control.mod (速度控制器)";
echo "  CN5: DIF_control.mod (差分控制器)";
echo "  CN6: POS_control.mod (位置控制器)";
echo " ";
echo " 示波器：ch2=位置回授，ch3=速度回授，ch4=扭力命令";
echo " ";                                                    //**畫面顯示結束

module .dsp.scope.ch1;                                       //**設定示波器 ch1
s addr &.fg2.out;  s zero 0; s y0 70; s dy 1000;
module .dsp.scope.ch2;                                       //**設定示波器 ch2
s addr &.io3.ang;  s zero 0; s y0 50; s dy 1000;
module .dsp.scope.ch3;                                       //**設定示波器 ch3
s addr &.cn5.dif;  s zero 0; s y0 30; s dy 1000;
module .dsp.scope.ch4;                                       //**設定示波器 ch4
s addr &.cn4.out;  s zero 0; s y0 10; s dy 1000;
module .dsp.scope;                                           //**設定示波器觸發和 X 軸
s chn  &.fg2.out;  s th 100; s x0 0;  s dx 15000; s mode 9; s trig 2;

module .io4;                                                 //**設定 io4 模組
s spd  &.io3.spd;
module .fg2;                                                 //**設定 fg2 模組
s kAmp 1000; s kFrq 15; s kDuty 128;
module .cn1;                                                 //**設定 cn1 模組
s ang &.io4.ang; s tq &.cn4.out; s kBias 0;
module .cn2;                                                 //**設定 cn2 模組
s cmd &.cn1.vcmd; s fdk &.io2.vo;
module .cn3;                                                 //**設定 cn3 模組
s cmd &.cn1.wcmd; s fdk &.io2.wo;
module  .cn4;                                                //**設定 cn4 模組
s cmd &.cn6.out;  s fdk &.cn5.dif;
module  .cn5;                                                //**設定 cn5 模組
s inp &.io3.ang;
module  .cn6;                                                //**設定 cn5 模組
```

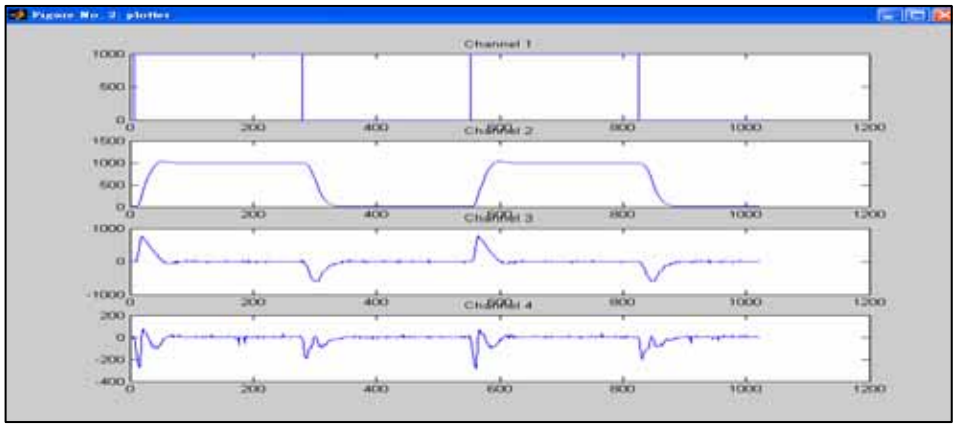
```
s cmd &.fg2.out; s fdk &.io3.ang;
module .io5;                                    //**設定 io5 模組
s vcmd &.cn2.out; s wcmd &.cn3.out;
module .ft1;                                    //**設定 ft1 模組
s u   &.fg2.out;

module .;                                       //**回復預設模組
run 8khz;                                       //**設定抽樣頻率為 4KHz
run start;                                      //**DSP 開始執行
echo on;                                        //**回復正常訊息顯示
```

對應之 Simulink mdl 檔說明：(exp5_angle.mdl)



圖二十二－ 實驗 5. 閉環路的位置控制

實測響應圖：



圖二十三－閉環路的位置控制響應圖

#6.加減速控制與前饋補償設計

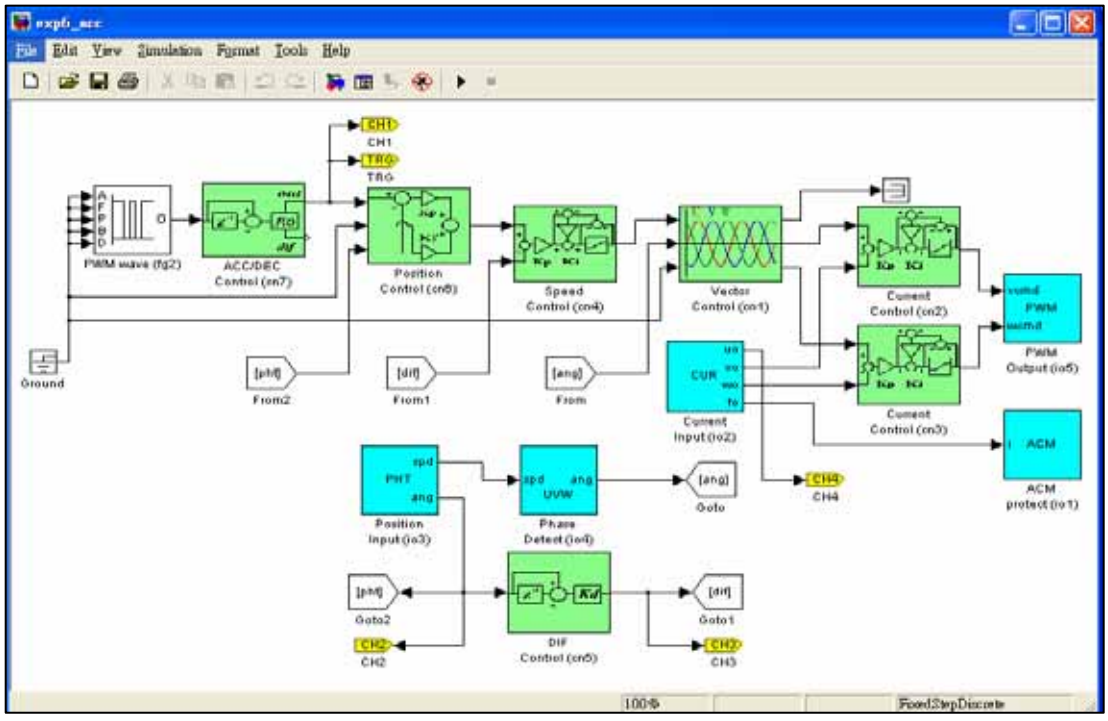
實驗說明：為避免方波命令的過於嚴苛，在方波命令的輸出，加入一加減速控制功能方塊，以減緩命令變化的幅度，增加馬達運動的平順。

對應之 plc 檔程式說明：(exp6_acc.plc)

```
echo off;                                                    //**畫面顯示開始
run stop;                                                    //**既有程式停止
run testACM;                                                 //**下載指定程式
prog motor.plc;                                              //**下載馬達參數
echo clear;                                                  //**畫面清除
echo "**********************************";
echo "*  實驗六: 加減速控制和前饋補償   *";
echo "**********************************";
echo " ";
echo "模組使用狀況：";
echo " ";
echo "  FG2: PWM_wave.mod    (方波產生器)";
echo "  CN1: VEC_control.mod (三相分配電路)";
echo "  CN2: CUR_control.mod (V 相電流控制)";
echo "  CN3: CUR_control.mod (W 相電流控制)";
echo "  CN4: SPD_control.mod (速度控制器)";
echo "  CN5: DIF_control.mod (差分控制器)";
echo "  CN6: POS_control.mod (位置控制器)";
echo "  CN7: ACC_control.mod (加減速控制器)";
echo " ";
echo " 示波器：ch2=位置回授，ch3=速度回授，ch4=扭力命令";
echo " ";                                                    //**畫面顯示結束

module .dsp.scope.ch1;                                       //**設定示波器 ch1
s addr &.cn7.out;  s zero 0; s y0 70; s dy 1000;
module .dsp.scope.ch2;                                       //**設定示波器 ch2
s addr &.io3.ang;  s zero 0; s y0 50; s dy 1000;
module .dsp.scope.ch3;                                       //**設定示波器 ch3
s addr &.cn5.dif;  s zero 0; s y0 30; s dy 1000;
module .dsp.scope.ch4;                                       //**設定示波器 ch4
s addr &.cn4.out;  s zero 0; s y0 10; s dy 1000;
module .dsp.scope;                                           //**設定示波器觸發和 X 軸
s chn  &.fg2.out;  s th 100; s x0 0;  s dx 15000; s mode 9; s trig 2;
module .io4;                                                 //**設定 io4 模組
s spd  &.io3.spd;

module .fg2;                                                 //**設定 fg2 模組
s kAmp 1000; s kFrq 15; s kDuty 128;
module .cn1;                                                 //**設定 cn1 模組
s ang &.io4.ang; s tq &.cn4.out; s kBias 0;
module .cn2;                                                 //**設定 cn2 模組
s cmd &.cn1.vcmd; s fdk &.io2.vo;
module .cn3;                                                 //**設定 cn3 模組
s cmd &.cn1.wcmd; s fdk &.io2.wo;
```

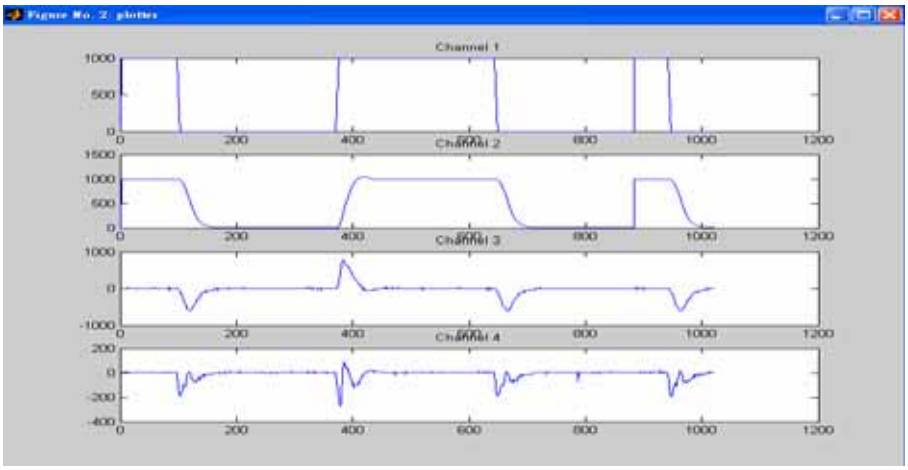
```
module  .cn4;                                          //**設定 cn4 模組
s cmd &.cn6.out;  s fdk &.cn5.dif;
module  .cn5;                                          //**設定 cn5 模組
s inp &.io3.ang;
module  .cn6;                                          //**設定 cn6 模組
s cmd &.cn7.out; s fdk &.io3.ang;
module  .cn7;                                          //**設定 cn7 模組
s cmd &.fg2.out;
module .io5;                                           //**設定 io5 模組
s vcmd &.cn2.out; s wcmd &.cn3.out;
module .ft1;                                           //**設定 ft1 模組
s u   &.fg2.out;

module .;                                              //**回復預設模組
run 8khz;                                              //**設定抽樣頻率為 4KHz
run start;                                             //**DSP 開始執行
echo on;                                               //**回復正常訊息顯示
```

對應之 simulink mdl 檔說明：(exp6_acc.mdl)



圖二八－ 實驗 6. 加減速控制和前饋補償

實測響應圖：



圖二十五－加減速控制和前饋補償響應圖

10. 模組程式說明

實驗中所使用的模組程式包括：

SIN_wave.asm ：正弦波產生器。
PWM_wave.asm ：方波產生器。
SAW_wave.asm ：鋸齒波產生器。
IIR1_filter.asm ：一階濾波器。
DAS4.asm ：資料蒐集模組。

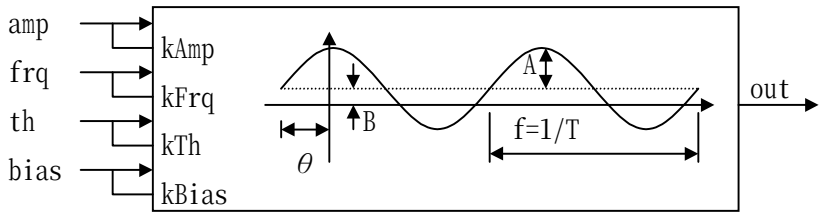
ACM_protect.asm：驅動級的保護界面。
PWM_output.asm ：三相 PWM 驅動界面。
CUR_input.asm ：三相電流回授界面。
POS_input.asm ：位置和速度回授界面。
UVW_input.asm ：馬達相角回授界面。

ACC_control.asm：加減速控制器。
POS_control.asm：位置迴路控制器(包括前饋補償)。
SPD_control.asm：速度迴路控制器(PI 控制器)。
CUR_control.asm：電流迴路控制器(PI 控制器)。
DIF_control.asm：差分控制器(速度回授計算)。
VEC_control.asm：相位分配控制器。

下面將針對交流伺服實驗專用的模組作詳細說明。

弦波產生器

此模組會產生週期性的正弦波。方塊圖如下：



其中：

1.正弦波有四個參數：振幅(A)、頻率(F)、相位(θ)和偏移(B)。其公式為：

$$out = A \sin(2\pi F t + \theta) + B$$

2.對於單純的正弦波產生器，所對映的參數包括：

kAmp ：振幅值。
kFrq ：頻率值，kFrq=65536/8000*Hz，其中 Hz 的範圍為 0~4000。例如
1000Hz => frq=8192
100Hz => frq=819
10Hz => frq=82

1Hz => frq=8

kTh ：相角值，0~255 代表 0~360 的相位。當
th=0 時為正弦波，
th=64 時為餘弦波，
th=128 時為反相波，依此類推。

kBias：偏移值。

3.對於可調變的正弦波產生器，其對映的輸入點為：

amp ：控制振幅的輸入點，可產生調幅(AM)波形。

frq ：控制頻率的輸入點，可產生調頻(FM)波形。

th ：控制相位的輸入點，可產生調相(PM)波形。

bias：控制偏移的輸入點，可產生雙頻混波(相加)波形。

4.若不使用調變功能時，各個輸入點時的預設值為

amp ：指向 kAmp 的位址。

frq ：指向 kFrq 的位址。

th ：指向 kTh 的位址。

bias：指向 kBias 的位址。

模組程式範例：SIN_WAVE.asm

```
@local  amp    R0   IP kAmp;                    //輸入點，sin 波的振幅
@local  frq    R1   IP kFrq;                    //輸入點，sin 波的頻率
@local  th     R2   IP kTh;                     //輸入點，sin 波的相角
@local  bias   R3   IP kBias;                   //輸入點，sin 波的中心偏移
@local  out    R4   OP;                         //輸出點，sin 波輸出
@local  tab    R5   TP;                         //表格指標，指向 sin[256]的表格
@local  ang    R6;                              //內部變數，相位角，65536=360°
@local  kAmp   R7   1000;                       //參數，振幅值
@local  kFrq   R8   1000;                       //參數，頻率值=65536/8000*Hz
@local  kTh    R9   0;                          //參數，相角值，360°=256
@local  kBias R10 0;                            //參數，偏移值
@data   11;                                     //模組變數區長度及初值
@initial:                                       //啟動程序
      WR ang,#0; WR out,#0;                     //ang=out=0
      RET;
@timer:                                         //即時控制程序(8KHz)
      RD  ang;    ADD (frq); WR ang;            //ang+=frq
      SFR 8;      ADD (th);  AND #255;          //ACC=sin(ang/256+th)
      ADD tab;    WR.H;      RD.M;              //table-look-up
      MUL (amp); SFR 12;;    ADD (bias); WR out; //out=ACC*amp/4096+bias
      RET;
@table tab 256;                                 //sin[256]表格區，1.0=4096
```

```
{ 0,     101,   201,   301,   401,   501,   601,   700   }; //00~07，0°
{ 799,   897,   995,   1092,  1189,  1285,  1380,  1474  }; //08~0F
{ 1567,  1660,  1751,  1842,  1931,  2019,  2106,  2191  }; //10~17
{ 2276,  2359,  2440,  2520,  2598,  2675,  2751,  2824  }; //18~1F
{ 2896,  2967,  3035,  3102,  3166,  3229,  3290,  3349  }; //20~27
{ 3406,  3461,  3513,  3564,  3612,  3659,  3703,  3745  }; //28~2F
{ 3784,  3822,  3857,  3889,  3920,  3948,  3973,  3996  }; //30~37
{ 4017,  4036,  4052,  4065,  4076,  4085,  4091,  4095  }; //38~3F
{ 4096,  4095,  4091,  4085,  4076,  4065,  4052,  4036  }; //40~47，90°
{ 4017,  3996,  3973,  3948,  3920,  3889,  3857,  3822  }; //48~4F
{ 3784,  3745,  3703,  3659,  3612,  3564,  3513,  3461  }; //50~57
{ 3406,  3349,  3290,  3229,  3166,  3102,  3035,  2967  }; //58~5F
{ 2896,  2824,  2751,  2675,  2598,  2520,  2440,  2359  }; //60~67
{ 2276,  2191,  2106,  2019,  1931,  1842,  1751,  1660  }; //68~6F
{ 1567,  1474,  1380,  1285,  1189,  1092,  995,   897   }; //70~77
{ 799,   700,   601,   501,   401,   301,   201,   101   }; //78~7F
{ 0,     65435, 65335, 65235, 65135, 65035, 64935, 64836 }; //80~87，180°
{ 64737, 64639, 64541, 64444, 64347, 64251, 64156, 64062 }; //88~8F
{ 63969, 63876, 63785, 63694, 63605, 63517, 63430, 63345 }; //90~97
{ 63260, 63177, 63096, 63016, 62938, 62861, 62785, 62712 }; //98~9F
{ 62640, 62569, 62501, 62434, 62370, 62307, 62246, 62187 }; //A0~A7
{ 62130, 62075, 62023, 61972, 61924, 61877, 61833, 61791 }; //A8~A8
{ 61752, 61714, 61679, 61647, 61616, 61588, 61563, 61540 }; //B0~B7
{ 61519, 61500, 61484, 61471, 61460, 61451, 61445, 61441 }; //B8~BF
{ 61440, 61441, 61445, 61451, 61460, 61471, 61484, 61500 }; //C0~C7，270°
{ 61519, 61540, 61563, 61588, 61616, 61647, 61679, 61714 }; //C8~CF
{ 61752, 61791, 61833, 61877, 61924, 61972, 62023, 62075 }; //D0~D7
{ 62130, 62187, 62246, 62307, 62370, 62434, 62501, 62569 }; //D8~DF
{ 62640, 62712, 62785, 62861, 62938, 63016, 63096, 63177 }; //E0~E7
{ 63260, 63345, 63430, 63517, 63605, 63694, 63785, 63876 }; //E8~EF
{ 63969, 64062, 64156, 64251, 64347, 64444, 64541, 64639 }; //F0~F7
{ 64737, 64836, 64935, 65035, 65135, 65235, 65335, 65435 }; //F8~FF
```

就程式結構而言，說明如下：

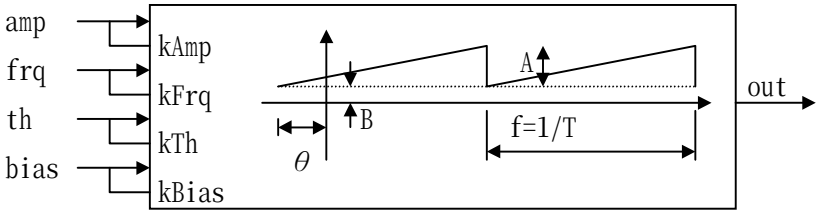
```
@timer:                                          //即時控制程序(8KHz)
    RD   ang;    ADD (frq); WR ang;              //ang+=frq
    SFR 8;       ADD (th);  AND #255;            //ACC=sin(ang/256+th)
    ADD tab;     WR.H;      RD.M;                //table-look-up
```

```
MUL (amp); SFR 12;;      ADD (bias); WR out; //out=ACC*amp/4096+bias
RET;
```

1.基本上我們以查表法計算 sin()。其輸入θ的範圍為 0~255，對映到 0~360°；其輸出 sin(θ)的範圍為+/-4096，對映到+/-1.0 的範圍。
2.建立一個內部變數 ang，其範圍為 0~65535，對映到 0~360 的相位角。
3.每個@timer 程序都執行 ang+=frq 的動作，更新目前的相位角。接著依照 ang 的相位角，查表得到 sin(ang/256+th)值。
4.由於表格長度為 256-word，所以必須將 ang/256 才能得到正確的對映關係。而 th 範圍為 0~255，所以不需要像 ang 一般的做額外處理。
5.sin()的範圍為(+1,-1)，而在表格中以(+4096~-4096)的型式記錄，相當於 1.0=4096。所以執行 A*sin()時還要用 SFR12 指令做/4096 的動作，才能正確的取得輸出值。
6.就頻率輸入 frq 而言，其計算原則是將$\theta=\omega t$轉化成$\theta=\Sigma(\omega*dt)$，因為抽樣頻率為 8KHz，所以 dt=1/8000。而得到 frq=65536/8000*Hz 公式，其中 Hz 為真正的頻率。在 8KHz 的抽樣頻率下，其範圍為 0~4000。

鋸齒波產生器

此模組會產生週期性的鋸齒波。方塊圖如下：



其中：

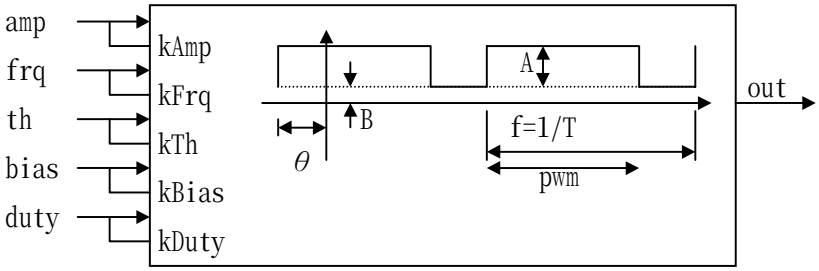
1.鋸齒波也有四個參數：振幅(A)、頻率(F)、相位(θ)和偏移(B)，和正弦波的定義完全一致。唯一的差異是其輸出為鋸齒波而非正弦波。
2.基本波形是從週期開始時為 0，一直遞增到週期結束時到達最高點，然後立刻回歸到 0 重新開始。
3.以程式結構而論，鋸齒波程式幾乎就是正弦波的相位計算部分，只差沒有正弦波的查表處理而已。

模組程式範例：SAW_WAVE.asm

```
@local   amp     R0    IP kAmp;                  //輸入點，sin 波的振幅
@local   frq     R1    IP kFrq;                  //輸入點，sin 波的頻率
@local   th      R2    IP kTh;                   //輸入點，sin 波的相角
@local   bias    R3    IP kBias;                 //輸入點，sin 波的中心偏移
@local   out     R4    OP;                       //輸出點，sin 波輸出
@local   ang     R5;                             //內部變數，相位角，65536=360°
@local   kAmp    R6    1000;                     //參數，振幅值
```

```
@local   kFrq   R7   1000;                                  //參數，頻率值=65536/8000*Hz
@local   kTh    R8   0;                                     //參數，相角值，360°=256
@local   kBias R9   0;                                      //參數，偏移值
@data    11;                                                //模組變數區長度及初值
@initial:                                                   //啟動程序
      WR ang,#0; WR out,#0;                                 //ang=out=0
      RET;
@timer:                                                     //即時控制程序(8KHz)
      RD   ang;     ADD (frq); WR ang;                      //ang+=frq
      SFR 8;        ADD (th);   AND #255;                   //ACC=(ang/256+th)&0x00ff
      MUL (amp); SFR 8;;      ADD (bias); WR out; //out=ACC*amp/256+bias
      RET;
```

脈衝波產生器



此模組會產生週期性的脈衝波。方塊圖如下：

其中：

1.脈衝波有五個參數，其中振幅(A)、頻率(F)、相位(θ)和偏移(B)四個參數和正弦波定義完全一樣，另外再增加一個負載率(D)。

2.基本波形是從週期開始時為 1，隔段時間後換為 0，然後持續到週期結束再重新開始。其中 1 所佔有的比例就是負載率，當 D=0 時為 0%，而 D=255 時為 100%。

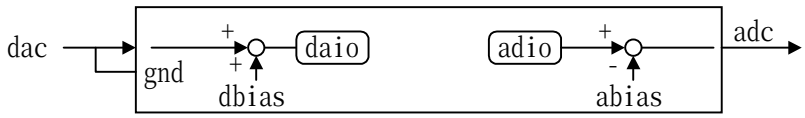
3.和其他參數類似，負載率(D)以 duty 為其輸入點，而以 kDuty 為其參數值。

模組程式範例：PLS_WAVE.asm

```
@local   amp     R0    IP kAmp;                             //輸入點，脈衝波的振幅
@local   frq     R1    IP kFrq;                             //輸入點，脈衝波的頻率
@local   th      R2    IP kTh;                              //輸入點，脈衝波的相角
@local   bias    R3    IP kBias;                            //輸入點，脈衝波的中心偏移
@local   duty    R4    IP kDuty;                            //輸入點，脈衝波的負載率
@local   out     R5    OP;                                  //輸出點，脈衝波輸出
```

```
@local   ang     R6;                                  //內部變數，相位角，65536=360°
@local   kAmp    R7   1000;                           //參數，振幅值
@local   kFrq    R8   1000;                           //參數，頻率值=65536/8000*Hz
@local   kTh     R9   0;                              //參數，相角值，360°=256
@local   kBias R10 0;                                 //參數，偏移值
@local   kDuty R11 128;                               //參數，負載率，100% =256
@data    12;                                          //模組變數區長度及初值
@initial:                                             //啟動程序
        WR ang,#0; WR out,#0;                         //ang=out=0
        RET;
@timer:                                               //即時控制程序(8KHz)
        RD   ang;       ADD (frq); WR ang;            //ang+=frq
        SFR 8;          ADD (th);   AND #255;         //ACC=(ang/256+th)&0x00ff
        SUB (duty); RD   (amp); JLT t1; RD #0;        //ACC=(amp,0)
t1:   ADD (bias); WR     out;                         //out=ACC+bias
        RET;
```

類比界面模組



此模組建立一組類比輸入輸出界面。方塊圖如下：

其中：

1.類比輸出界面是將 dac 輸入點的訊號送入 daio 的類比輸出阜中。如此就可將其他模組的輸出訊號送至類比輸出端。

2.類比輸入界面是將 adio 的類比輸入界面，送至 adc 的輸出端。如此就可讓其他模組順利取得 adc 的讀值。

3.在類比輸出時還會加上偏壓，公式為

(daio) = (dac) + dbias

4.在類比輸入時還會減掉偏壓，公式為

adc = (adio) - abias

以數位通訊實驗的觀點而言，ADC 和 DAC 的實際位址對映如下：

ADC1=0x40	LCD 模組的電話輸入端	DAC1=0x70	LCD 模組的電話輸出端
ADC2=0x41	EXP 模組的電話輸入端	DAC2=0x72	EXP 模組的電話輸出端

模組程式範例：ADDA.asm

```
@local     adc      R0    OP;                          //輸出點，ADC 的讀值
@local     dac      R1    IP gnd;                      //輸入點，DAC 的命令
@local     adio     R2    0x40;                        //IO 界面，ADC 的絕對位址
@local     daio     R3    0x70;                        //IO 界面，DAC 的絕對位址
@local     abias R4    2000;                          //內部變數，ADC 的零點
@local     dbias R5    2400;                          //內部變數，DAC 的零點
@local     gnd      R6    0;                           //內部變數，相當於接地點
@data      7;                                         //模組變數區長度=13
@initial:                                             //*****啟動程序*****
@stop:     RD dbias; WR (daio); RET;                  //*****結束程序*****
@timer:                                               //*****即時控制*****
        RD (adio); AND #0xfff; SUB abias; WR    adc;       //ADC 輸入處理
        RD (dac);                  ADD dbias; WR (daio);   //DAC 輸出處理
        RET;                                                  //結束
```

資料蒐集模組

資料蒐集模組是一個非常特殊的模組，專門配合 eTTY 程式中的示波器使用。

模組程式範例：DAS4.asm

```
@global th      R12;
@global trg     R13;
@global ch1     R16;
@global ch2     R20;
@global ch3     R24;
@global ch4     R28;
@local     cmd     R0 0;     //紀錄命令，0=停止,1=前置記錄,2=正觸發,4=負觸發,6=觸發記錄
@local     dt      R1 1;     //抽樣週期，每 dT 抽樣一次
@local     tab     R2 TP;    //DAS 表格啟始位址
@local     cnt     R3 0;     //目前記錄點(0~1023)
@local     buf     R4 1;     //抽樣計數，(1~dt)
@local     pnt     R5;       //表格位址計算，pnt=tab+cnt*4
@local     len     R6;       //表格長度計算(0~1023)
@data              7;        //定義模組變數區的長度
@table     tab 4096;         //定義表格的長度
@initial:                    //*****啟動程序*****
```

```
        WR cmd,#0; RET; //cmd=0
@timer:                         //*****控制程序*****
        ADD cmd,#0; JZ    t9;                       //if (cmd==0)      return;
        SUB buf,#1; JNZ t9;                         //if (--buf)>0) return;
        RD dt;    WR buf;                           //dt=buf;
        RD cnt; ADD #1;    AND #0x3ff; WR cnt; //cnt=(cnt+1)&0x3ff;
        SFL 2;    ADD tab; WR pnt;                  //pnt=tab+cnt*4;
        RD.0 (ch1); WR (pnt); ADD pnt,#1;       //*pnt++=*ch1;
        RD.0 (ch2); WR (pnt); ADD pnt,#1;       //*pnt++=*ch2;
        RD.0 (ch3); WR (pnt); ADD pnt,#1;       //*pnt++=*ch3;
        RD.0 (ch4); WR (pnt);                     //*pnt    =*ch4;
        RD cmd; SUB #1; JZ t9; SUB #1; JZ t2; SUB #1; JZ t3;
                SUB #1; JZ t4; SUB #1; JZ t5; SUB #1; JZ t6; JR t7;
t2:    RD.0 (trg); SUB.0 th; JGE t9; ADD cmd,#1; JR t9; //wait *trg<th
t3:    RD.0 (trg); SUB.0 th; JLE t9; JR t6;                  //wait *trg>th
t4:    RD.0 (trg); SUB.0 th; JLE t9; ADD cmd,#1; JR t9; //wait *trg>th
t5:    RD.0 (trg); SUB.0 th; JGE t9; JR t6;                  //wait *trg<th
t6:    WR    len,#0; WR    cmd,#7;                             //len=0;cmd=7
t7:    ADD len,#1; SUB #0x200; JLT t9; WR cmd,#0;          //if ((++len)>=0x200) cmd=0
t9:    RET;
```

驅動級保護模組

此模組處理整個功率驅動級的管理和保護工作。方塊圖如下:

其中：

1.i 是馬達的電流量測值，用以確認馬達電流在安全範圍內。

2.開機後延遲幾個安全週期，之後才啟動看護犬的觸發工作。

3.參數 zone 設定功率驅動級的切換保護，範圍為 0~63，單位為 12MHz。

4.參數 enb 設定單軸的啟動控制，0=OFF、1=啟動。

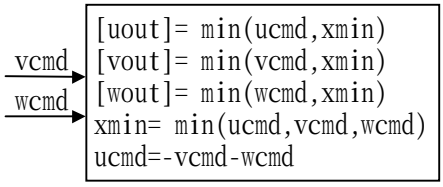
模組程式範例：ACM_protect.asm

```
@global seq    R0;                    //公用變數，@timer 程序的執行次數
@local    i       R0    IP gnd;        //輸入點，馬達電流量測值
@local    enb    R1    1;               //參數，0=OFF，1=ON
```

```
@local   pwm   R2   0x31;          //輸出阜地址，PWM 參數點(0x30/0x31)
@local   zone R3   60;            //參數，PWM 切換保護(0~63)，60=5us
@local   dly   R4;                //內部變數，啟動延遲計數
@local   gnd   R5   0;            //輸入短路點
@data    6;                       //模組變數區長度
@initial:                         //*****  開機啟動程序  *****
      WR dly,#10; RD zone; WR (pwm); RET;          //dly=10,pwm=zone
@timer:                           //*****  即時控制程序  *****
      RD dly;    SUB #1;   JN b0; WR dly;   JR b9;      //dly--
b0:   RD.0 seq; AND enb; SFL 7; OR zone; WR (pwm); //pwm=((enb&seq)<<7)+zone
b9:   RET;
```

PWM 電壓輸出模組

此模組產生 PWM 型式的等效電壓輸出，方塊圖如下：



其中：

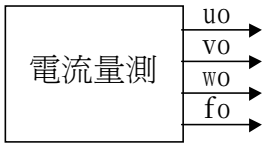
1. 因為 ucmd+vcmd+wcmd=0，所以只需要兩相命令即可產生三相電壓。
2. 先取得最低電壓 xmin=min(ucmd,vcmd,wcmd)，再分別計算
   uout = min(ucmd,xmin)
   vout = min(vcmd,xmin)
   wout = min(wcmd,xmin)
   所以三相 PWM 命令中最少有一相的 PWM 命令為 0。
3. 只有 PWM>0 時才需要高頻的切換動作，所以 PWM=0 的輸出完全不需要切換，如此可大幅降低功率晶體的能量消耗。

模組程式範例：PWM_output.asm

```
@local   vcmd   R0   IP gnd;          //輸入點，v 相命令
@local   wcmd   R1   IP gnd;          //輸入點，w 相命令
@local   ucmd   R2;                   //內部變數，u 相命令
@local   uout   R3   0x53;            //輸出阜地址，u 相 PWM 輸出(0x50/0x53)
@local   vout   R4   0x54;            //輸出阜地址，v 相 PWM 輸出(0x51/0x54)
@local   wout   R5   0x55;            //輸出阜地址，w 相 PWM 輸出(0x52/0x55)
@local   lmtH   R6   3072;            //常數，PWM 輸出上限
@local   xmin   R7;                   //內部變數，三相命令最低值
@local   gnd    R8   0;               //輸入接地點
```

```
@data     9;                                    //模組變數區長度
@initial:                                       //***** 開機啟動程序 *****
      RD #0; WR (uout); WR (vout); WR (wout); RET;   //uout=vout=wout=0
@timer:                                         //***** 即時控制程序 *****
      RD (vcmd); ADD (wcmd); NEG;          WR ucmd;     //ucmd=-(vcmd+wcmd)
                  MIN (vcmd); MIN (wcmd); WR xmin;
//xmin=min(ucmd,vcmd,wcmd)
      RD  ucmd;  SUB  xmin;; MIN  lmtH;  WR (uout); //uout=lmt(ucmd-xmin,xmax)
      RD (vcmd); SUB  xmin;; MIN  lmtH;  WR (vout); //vout=lmt(ucmd-xmin,xmax)
      RD (wcmd); SUB  xmin;; MIN  lmtH;  WR (wout); //wout=lmt(ucmd-xmin,xmax)
      RET;
```

電流量測模組



此模組處理 ADC 輸入阜訊號，取得三相電流的量測值。方塊圖如下:

其中：

1.uo/vo/wo 是三相的電流值，而 fo 是保護計算用的平均電流值，其公式為

fo = (abs(uo) + abs(vo) + abs(wo)) /2

2.vo 和 wo 是由 ADC 電路直接量測，其公式為

vo = ((vi)-vz)*vk/4096

wo = ((wi)-wz)*wk/4096

其中參數包括

vi/wi：ADC 的對應位址，X 模組為(0x48/0x4c)，而 Y 模組為(0x49/0x4d)。

vz/wz：當電流為 0 時的 ADC 讀值

vk/wk：ADC 的對應增益，單位為 1.0=4096

3.uo 是由其他兩相電流計算而得，因為 uo+vo+wo=0，所以 uo=-(vo + wo)

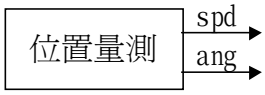
模組程式範例：CUR_input.asm

```
@local   uo     R0    OP;          //輸出點，u 相電流量測值
@local   vo     R1    OP;          //輸出點，v 相電流量測值
@local   wo     R2    OP;          //輸出點，w 相電流量測值
@local   fo     R3    OP;          //輸出點，平均輸出電流(保護用)
@local   vi     R4    0x49;        //輸入阜地址，v 相電流輸入點(0x48/0x49)
@local   wi     R5    0x4d;        //輸入阜地址，w 相電流輸入點(0x4c/0x4d)
```

```
@local   vz     R6   2048;                    //v 相輸入零點，2048=50%
@local   wz     R7   2048;                    //w 相輸入零點，2048=50%
@local   vk     R8   4096;                    //v 相輸入增益，1.0=4096
@local   wk     R9   4096;                    //w 相輸入增益，1.0=4096
@data    10;                                  //模組變數區長度
@initial:                                     //***** 開機啟動程序 *****
      RD #0; WR uo; WR vo; WR wo; WR fo; RET;     //uo=vo=wo=fo=0
@timer:                                       //***** 即時控制程序 *****
      RD (vi); AND #0xfff; SUB vz; MUL vk; SFR 12; WR vo; //vo=(vi-vz)*vk/4096
      RD (wi); AND #0xfff; SUB wz; MUL wk; SFR 12; WR wo; //wo=(wi-wz)*wk/4096
      ADD vo;   NEG;                                      WR uo; //uo=-(vo+wo)
      ABS; RD.S vo; ABS; RD.S wo; ABS;              //fo=(abs(uo)+abs(vo)+abs(wo))/2
      ADD.S; ADD.S; SGN; SFR; WR fo; RET;
```

位置量測模組

此模組將光電盤的 A/B 相訊號轉換成速度和角度訊號輸出。方塊圖如下：



其中：

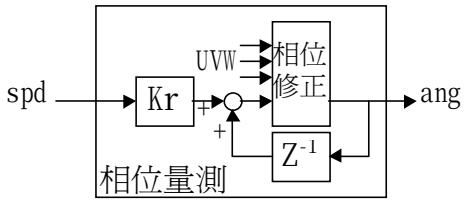
1. ang 是 16-bit 的角度量測值，其單位為 pulse。若光電盤的解析度為 2000PPR，則 ang 單位為(1/8000)圈。
2. spd 是 ang 的差分處理結果(即速度值)，其單位為 pulse/sample。若馬達速度為 1000RPM，而抽樣頻率為 4KHz，則 spd 值將為 1000*8000/60/4000=33。
3. 將 angH/ang 合併即可取得 32-bit 的角度量測值。不過在實驗中還是以 16-bit 的角度量測為主，32-bit 值只供參考用。
4. pht 為光電盤計數器的實際位址，X 模組為 0x68，而 Y 模組為 0x69。

模組程式範例：POS_input.asm

```
@local   spd     R0 OP;                       //輸出點，速度量測值
@local   ang     R1 OP;                       //輸出點，角度量測值
@local   angH    R2;                          //內部變數，角度的 32 位元部分
@local   pht     R3 0x6a;                     //輸入阜地址，光電盤輸入點(0x68/0x6a)
@local   old     R4;                          //內部變數，速度更新用
@data    5;                                   //模組變數區長度
@initial:                                     //***** 開機啟動程序 *****
      RD (pht); WR old; WR spd,#0; RET;            //old=pht,spd=0
@timer:                                       //***** 即時控制程序 *****
```

```
RD (pht); SUB.X old; SGN 11; WR spd;          //spd=(pht-old),old=pht
          ADD ang;     WR ang;                 //angHL+=spd
RD.H;     ADC angH;    WR angH;
RET;
```

相位量測模組



此模組處理光電盤的 UVW 相位訊號，計算馬達的絕對角度值。方塊圖如下:

其中：

1.UVW 相位是光電盤的絕對角量測，其量測精度為 60°。

2.為了取得更精密的量測結果，可再經由 spd 推算相對角部分。其公式如下：

ang.angL = ang.angL + spd*kr/65536

換句話說，馬達角度可由馬達速度的積分取得。

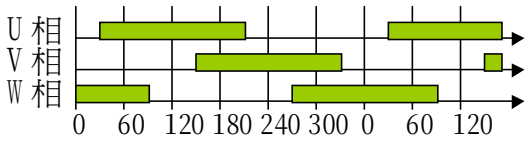
3.如果估測的馬達角度和 UVW 相位一致，則不作處理。否則必須以 UVW 相位為準，修正馬達角度至正確的 UVW 相位範圍。

4.其中 angL 是 ang 的小數部分，使得 ang 的計算精度可達 32-bit，以避免積分時的累積誤差。

5.馬達角度 ang 的範圍為 0~255，代表 0~360°的角度範圍。所以如果光電盤解析度為 2000PPR，則 Kr 的計算公式為

Kr = 256 / 8000 * 65536

= 2097



在光電盤的輸入訊號中，UVW 相位的排列如下：

其中：

1.UVW 各相都涵蓋 180°的範圍，而各相相差 120°。

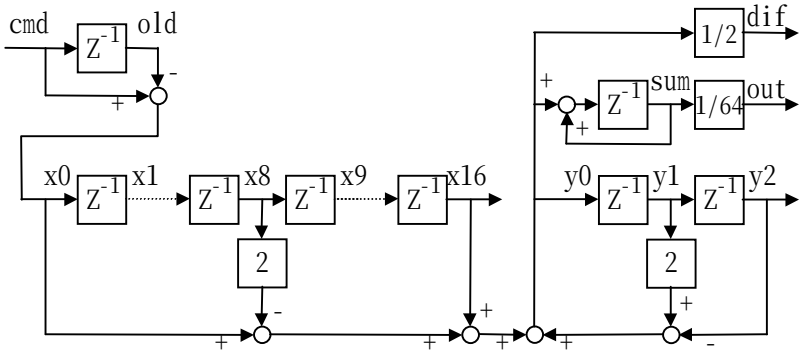
2.三個相位將馬達一圈區分成六組區域，每組涵蓋 60°範圍，依序為 30~90°、90~150°、150~210°、210~270°、270~330°、330~30°。

3.所以馬達角度的修正，就可依照表查表法取得。

模組程式範例：UVW_input.asm

```
@local   spd     R0   IP;                          //輸入點，速度量測值
@local   ang     R1   OP;                          //輸出點，角度量測值
@local   uvw     R2   0x61;                        //輸出阜地址，光電盤 UVW 輸入
(0x60/0x61)
@local   kr      R3   6711;                        //參數，增益 Kr=256*pole/2/PPR/4*65536
@local   tab     R4   TP;                          //表格指標，指向 UVW 對應表
@local   angL    R5;                               //內部變數，角度的 32 位元部分
@local   lmtH    R6;                               //內部變數，角度上限
@local   lmtL    R7;                               //內部變數，角度下限
@data    8;                                        //模組變數區長度
@initial:                                          //***** 開機啟動程序 *****
      WR ang,#0; WR angL,#0; RET;                  //angHL=0
@timer:                                            //***** 即時控制程序 *****
      RD (uvw); SFL 1; AND #14; ADD tab; WR.H;     //H=tab+uvw*2
      RD.M;       WR lmtL; RD.M;       WR lmtH;    //lmtL=(H++), lmtH=(H)
      RD (spd); MUL kr;   ADD angL; WR angL;       //angHL+=(spd*kr/4096)
      RD.H;       ADC ang; AND #255; WR ang;
      RD lmtL; ADD #0;    JZ   b9;                 //若 lmtL=0 則不處理
               SUB lmtH; JLE b1;                   //若 lmtL<lmtH 則區域#1~#5
      RD ang;   SUB #128; JGE b3;    JR b2;        //檢查區域#6
b1:   RD ang;   MAX lmtL; MIN lmtH; WR ang; RET; //區域#1~#5
b2:   RD ang;             MIN lmtH; WR ang; RET; //區域#6，ang<180°
b3:   RD ang;   MAX lmtL;           WR ang;      //區域#6，ang>180°
b9:   RET;
@table tab 16;                //***** 表格區，相角對映表 *****
      { 0,     0     };       //WVU=000,error
      { 64,    107 };        //WVU=001,區域#2: 90°~150°，256* 90/360=64
      { 149, 192 };          //WVU=010,區域#4:210°~270°，256*210/360=149
      { 107, 149 };          //WVU=011,區域#3:150°~210°，256*150/360=107
      { 235, 21   };         //WVU=100,區域#6:330°~30° ，256*330/360=235
      { 21,    64   };       //WVU=101,區域#1: 30°~90° ，256* 30/360=21
      { 192, 235 };          //WVU=110,區域#5:270°~330°，256*270/360=192
      { 0,     0     };      //WVU=111, error
```

加減速控制模組



此模組執行位置控制前的加減速控制。方塊圖如下：

伺服控制模組的控制頻率有兩種，電流迴路控制以 4KHz 的抽樣頻率執行，而位置和速度控制以 1KHz 的抽樣頻率執行。這種雙頻率的模組設計方式如下：

1. 設定一個模組參數 axis，指定馬達編號(0~3)，其中 0=X 軸，1=Y 軸，3=Z 軸，4=C 軸。
2. 公用變數 seq 是對映@timer 程序的執行次數，每次執行都會自動加一。
3. 當 seq 的最小兩位元和 axis 符合才會執行此模組，換句話說，每四次@timer 程序才會執行一次。程式如下：

```
RD.0 seq;; XOR axis;; AND #3;; JNZ b0;
.....實際控制程式.....
b9: RET;
```

4. 所以控制程式的抽樣頻率會從 4KHz 降到 1KHz。
5. 不但如此，當四軸同時控制時，因為各軸的 axis 參數定義不同，會自動錯開各軸的執行時間，相當於各軸輪流執行，使得@timer 程序的時間較為均勻。換句話說，同樣的 4KHz 週期中可以執行更多的模組程式。

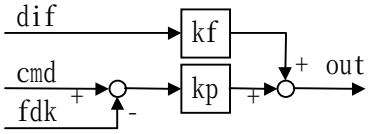
不只是加減速控制模組，所有抽樣頻率為 1KHz 的控制模組都會用到同樣的程式架構，屆時就不另加說明了。

模組程式範例：ACC_control.asm

```
@global   seq   R0;                          //公用變數，@timer 程序的執行次數
@local    cmd   R0   IP;                     //輸入點，位置命令
@local    out   R1   OP;                     //輸出點，加減速處理後的位置命令
@local    dif   R2   OP;                     //輸出點，位置微分命令
@local    axis R3   0;                       //馬達編號，0=X,1=Y,2=Z,3=C
@local    old   R4;                          //內部變數，微分計算用
@local    sumL R5;                           //內部變數，積分計算用
@local    sumH R6;                           //內部變數，積分計算用
@local x0   R7;                              //內部變數，x(0~16)
@local x1   R8;   @local x2   R9;   @local x3   R10; @local x4   R11;
@local x5   R12; @local x6   R13; @local x7   R14; @local x8   R15;
@local x9   R16; @local x10 R17; @local x11 R18; @local x12 R19;
@local x13 R20; @local x14 R21; @local x15 R22; @local x16 R23;
@local y0   R24;                             //內部變數，y(0~2)
@local y1   R25; @local y2   R26;
@data   27;                                  //模組變數區長度
@initial:                                    //***** 開機啟動程序 *****
      WR dif,#0; RD (cmd); WR out; WR old;          //dif=0, out=old=cmd, sumHL=cmd*64
      SFL 6; WR sumL; RD out; SFR 10; SGN 5; WR sumH;
      RPT #17; WR x0,#0; RPT #3; WR y0,#0; RET; //x[0~16]=0,y[0~2]=0
@timer:                                      //***** 即時控制程序 *****
      RD.0 seq; XOR axis;   AND #3; JNZ b9;         //1KHz 抽樣頻率
      RD (cmd); SUB.X old; WR x0;                    //x0=cmd-old, old=cmd
      SUB x8,1; ADD x16; ADD y1,1; SUB y2; WR y0;          //y0=x0-2x8+x16+2y1-y2
      SGN; ADD sumL; WR sumL; RD.H; ADC sumH; WR sumH; //sumHL+=y0
      WR.H; RD sumL; SFR 6; WR out;                  //out=sumHL/64
      RD y0;    SGN;   SFR;    WR dif;               //dif=y0*32/m/m
//       RD x0; RPT #16; RD.X x1;                    //x[i]=x[i-1], i=1~16
      RD x0; RD.X x1;   RD.X x2;   RD.X x3;   RD.X x4;   RD.X x5;   RD.X x6;
            RD.X x7;   RD.X x8;   RD.X x9;   RD.X x10; RD.X x11; RD.X x12;
            RD.X x13; RD.X x14; RD.X x15; WR    x16;
      RD y0; RD.X y1;   WR    y2;                  //y[i]=y[i-1], i=1~2
b9:   RET;
```

位置迴路控制模組

此模組執行比例控制器以及前饋補償，為位置迴路的控制器。方塊圖如下：



其中的輸入點定義方式較為特殊，說明如下：

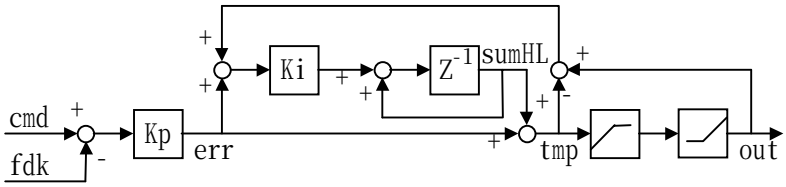
1. 一般的輸入點是定義要連接的輸出點位址，而以 RD(r) 指令讀入。這個位址本身在模組程式中並未設定初值，而要在系統程式中另行設定。
2. 所以如果系統程式中忽略此輸入點，就會造成無法預期的計算錯誤。
3. 此模組中共有三個輸入點，其中 cmd 是固定輸入點，一定要設定；而 dif 和 fdk 兩個輸入點是允許忽略的，當忽略時會自動讀入 0。
4. 因此我們在模組中特別規劃了一個 gnd 變數，固定為 0。而將 dif 和 fdk 的初值都指向 gnd 的位址。所以如果在系統程式中忽略了 dif 和 fdk，仍可正常工作。

模組程式範例：POS_control.asm

```
@global   seq   R0;                                  //公用變數，@timer 程序的執行計數
@local    cmd   R0   IP;                             //輸入點，位置命令
@local    dif   R1   IP gnd;                         //輸入點，位置微分命令(可忽略)
@local    fdk   R2   IP gnd;                         //輸入點，位置回授(可忽略)
@local    out   R3   OP;                             //輸出點，用以控制速度迴路
@local    axis R4   0;                               //馬達編號，0=X,1=Y,2=Z,3=C，
@local    kp    R5   4096;                           //參數,增益 kp，1.0=4096
@local    kf    R6   0;                              //參數,增益 kd，1.0=4096
@local    gnd   R7   0;                              //無輸入時的短路點
@data     8;                                         //模組變數區長度
@initial:                                            //***** 開機啟動程序 *****
      WR out,#0; RET;                                      //out=0
@timer:                                              //***** 即時控制程序 *****
      RD.0 seq; XOR axis;   AND #3; JNZ b9;                //1KHz 抽樣頻率模式
      RD (cmd); SUB (fdk); MUL kp; SFR 12;                 //ACC=(cmd-fdk)*kp/4096
      RD.S (dif); MUL kf;   SFR 12; ADD.S; WR out; //out=ACC+(dif*kd/4096)
b9:   RET;
```

速度迴路控制模組

此模組執行比例/積分控制器，並處理過溢狀況(overflow)的保護工作，適用於速度迴路的控制器。方塊圖如下：

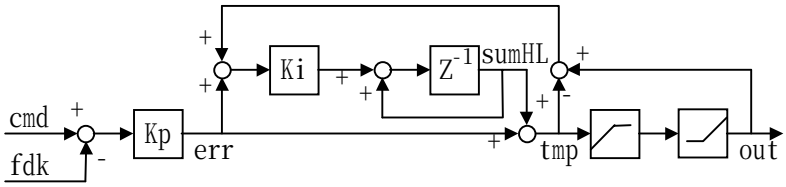


模組程式範例：SPD_control.asm

```
@global   seq   R0;                          //公用變數，@timer 程序的執行次數
@local    cmd   R0    IP   gnd;              //輸入點，速度命令
@local    fdk   R1    IP   gnd;              //輸入點，速度回授
@local    out   R2    OP;                    //輸出點，用以控制扭力迴路
@local    axis R3     0;                     //馬達編號，0=X,1=Y,2=Z,3=C
@local    kp    R4    4096;                  //參數,增益 KP，1.0=4096
@local    ki    R5    0;                     //參數,增益 KI，1.0=65536
@local    high R6     1000;                  //參數,輸出上限
@local    low   R7    -1000;                 //參數,輸出下限
@local    err   R8;                          //內部變數
@local    tmp   R9;                          //內部變數
@local    sumL R10;                          //內部變數
@local    sumH R11;                          //內部變數
@local    gnd   R12;                         //輸入接地點
@data     13;                                //模組變數區長度
@initial:                                    //*****  開機啟動程序  *****
      RD #0; WR out; WR sumL; WR sumH; RET;          //out=sumH=sumL=0
@timer:                                      //*****  即時控制程序  *****
      RD.0 seq; XOR axis;   AND #3; JNZ b9;          //1KHz 抽樣頻率模式
      RD (cmd); SUB (fdk); MUL kp; SFR 12; WR err; //err=(cmd-fdk)*kp/4096
      ADD sumH; WR   tmp;                            //tmp=err+sumH
      MIN high; MAX low;    WR out;                  //out=limit(tmp,high,low)
      SUB tmp;   ADD err;   MUL ki;                  //ACC=(out-tmp+err)*ki
                 ADD sumL;  WR sumL;                 //sumHL+=ACC
      RD.H;      ADC sumH;  WR sumH;
b9:   RET;
```

電流迴路控制模組

此模組執行比例/積分控制器，並處理過溢狀況(overflow)的保護工作，適用於電流迴路的控制器。方塊圖如下：

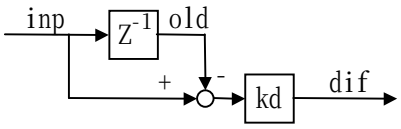


模組程式範例：CUR_control.asm

```
@local    cmd   R0    IP   gnd;          //輸入點，速度命令
@local    fdk   R1    IP   gnd;          //輸入點，速度回授
@local    out   R2    OP;                //輸出點，用以控制扭力迴路
@local    kp    R3    4096;              //參數,增益 KP，1.0=4096
@local    ki    R4    0;                 //參數,增益 KI，1.0=65536
@local    high R5     1000;              //參數,輸出上限
@local    low   R6    -1000;             //參數,輸出下限
@local    err   R7;                      //內部變數
@local    tmp   R8;                      //內部變數
@local    sumL R9;                       //內部變數
@local    sumH R10;                      //內部變數
@local    gnd   R11;                     //輸入接地點
@data     12;                            //模組變數區長度
@initial:                                //***** 開機啟動程序 *****
      RD #0; WR out; WR sumL; WR sumH; RET;          //out=sumH=sumL=0
@timer:                                  //***** 即時控制程序 *****
      RD (cmd); SUB (fdk); MUL kp; SFR 12; WR err; //err=(cmd-fdk)*kp/4096
      ADD sumH; WR   tmp;                          //tmp=err+sumH
      MIN high; MAX low;    WR out;                //out=limit(tmp,high,low)
      SUB tmp;   ADD err;   MUL ki;                //ACC=(out-tmp+err)*ki
                 ADD sumL;  WR sumL;               //sumHL+=ACC
      RD.H;      ADC sumH;  WR sumH;
b9:   RET;
```

差分計算模組

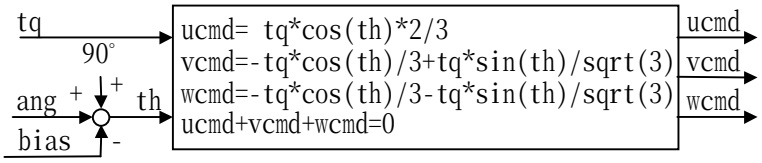
此模組對輸入訊號做差分，可做為速度回授的計算。方塊圖如下：



模組程式範例：DIF_control.asm

```
@global   seq   R0;                              //公用變數，@timer 程序的執行次數
@local    inp   R0    IP;                        //輸入點，訊號輸入
@local    dif   R1    OP;                        //輸出點，差分輸出
@local    axis R2     0;                         //馬達編號，0=X,1=Y,2=Z,3=C
@local    kd    R3    256;                       //差分增益，1.0=256
@local    old   R4;                              //內部暫存器，差分計算用
@data     5;                                     //模組變數區長度
@initial:                                        //*****  開機啟動程序  *****
      WR dif,#0; RD (inp); WR old; RET;          //dif=0,old=inp
@timer:                                          //*****  即時控制程序  *****
      RD.0 seq; XOR axis;   AND #3; JNZ b9;      //1KHz 抽樣頻率模式
      RD (inp); SUB.X old;                       //ACC=inp-old, old=inp
      MUL kd;     SFR 8;        WR dif;          //dif=ACC*Kd/256
b9:   RET;
```

相位分配模組



此模組將扭力命令轉換成三相的電流命令。方塊圖如下：

模組程式範例：VEC_control.asm

```
@local    tq      R0    IP kTq;              //輸入點，扭力命令，1.0=4096(max)
@local    ang     R1    IP kAng;             //輸入點，馬達轉角，360°=256
@local    bias    R2    IP kBias;            //輸入點，轉角偏移，360°=256
@local    ucmd    R3    OP;                  //輸出點，U 相電流，1Amp=4096
@local    vcmd    R4    OP;                  //輸出點，V 相電流，1Amp =4096
@local    wcmd    R5    OP;                  //輸出點，W 相電流，1Amp =4096
@local    tab     R6    TP;                  //表格指標，指向 sin[]表格
@local    th      R7;                        //內部變數，角度計算值
@local    sin     R8;                        //內部變數， sin(th)
@local    cos     R9;                        //內部變數，-cos(th)
@local    kd3     R10 1931;                  //常數，sqrt(2)/3          *4096=1931
@local    ks3     R11 3344;                  //常數，sqrt(2)/sqrt(3)*4096=3344
@local    kTq     R12 0;                     //輸入值，扭力命令，1.0=4096
@local    kAng    R13 0;                     //輸入值，馬達轉角，360°=256
@local    kBias R14 0;                       //輸入值，轉角偏移，360°=256
@data     15;                                //模組變數區長度
@initial:                                    //***** 開機啟動程序 *****
      WR ucmd,#0; WR vcmd,#0; WR wcmd,#0;    //ucmd=vcmd=wcmd=0
      RET;
@timer:                                      //***** 即時控制程序 *****
      RD (ang); SUB (bias); ADD #64; WR th;  //th=ang-bias+90°
      ADD #64; AND #255; ADD tab; WR.H;      //cos=-cos(th)*(1/3)
      RD.M; MUL kd3; SFR 12; NEG; WR cos;
      RD th;    AND #255; ADD tab; WR.H;     //sin=sin(th)*(1/sqrt(3))
      RD.M; MUL ks3; SFR 12;       WR sin;
```

```
    RD cos; ADD sin; MUL (tq); SFR 12; WR vcmd; //vcmd=tq*(cos+sin)
    RD cos; SUB sin; MUL (tq); SFR 12; WR wcmd; //wcmd=tq*(cos-sin)
            ADD vcmd;           NEG;    WR ucmd; //ucmd=-vcmd-wcmd
    RET;

@table tab 256;                         //***** sin[256] 表格區 *****
    { 0,    101,    201,    301,    401,    501,    601,    700     }; //00~07，0°
    { 799,  897,    995,    1092,   1189,   1285,   1380,   1474    }; //08~0F
    { 1567, 1660,   1751,   1842,   1931,   2019,   2106,   2191    }; //10~17
    { 2276, 2359,   2440,   2520,   2598,   2675,   2751,   2824    }; //18~1F
    { 2896, 2967,   3035,   3102,   3166,   3229,   3290,   3349    }; //20~27
    { 3406, 3461,   3513,   3564,   3612,   3659,   3703,   3745    }; //28~2F
    { 3784, 3822,   3857,   3889,   3920,   3948,   3973,   3996    }; //30~37
    { 4017, 4036,   4052,   4065,   4076,   4085,   4091,   4095    }; //38~3F
    { 4096, 4095,   4091,   4085,   4076,   4065,   4052,   4036    }; //40~47，90°
    { 4017, 3996,   3973,   3948,   3920,   3889,   3857,   3822    }; //48~4F
    { 3784, 3745,   3703,   3659,   3612,   3564,   3513,   3461    }; //50~57
    { 3406, 3349,   3290,   3229,   3166,   3102,   3035,   2967    }; //58~5F
    { 2896, 2824,   2751,   2675,   2598,   2520,   2440,   2359    }; //60~67
    { 2276, 2191,   2106,   2019,   1931,   1842,   1751,   1660    }; //68~6F
    { 1567, 1474,   1380,   1285,   1189,   1092,   995,    897     }; //70~77
    { 799,  700,    601,    501,    401,    301,    201,    101     }; //78~7F
    { 0,    65435, 65335, 65235, 65135, 65035, 64935, 64836 }; //80~87，180°
    { 64737, 64639, 64541, 64444, 64347, 64251, 64156, 64062 }; //88~8F
    { 63969, 63876, 63785, 63694, 63605, 63517, 63430, 63345 }; //90~97
    { 63260, 63177, 63096, 63016, 62938, 62861, 62785, 62712 }; //98~9F
    { 62640, 62569, 62501, 62434, 62370, 62307, 62246, 62187 }; //A0~A7
    { 62130, 62075, 62023, 61972, 61924, 61877, 61833, 61791 }; //A8~A8
    { 61752, 61714, 61679, 61647, 61616, 61588, 61563, 61540 }; //B0~B7
    { 61519, 61500, 61484, 61471, 61460, 61451, 61445, 61441 }; //B8~BF
    { 61440, 61441, 61445, 61451, 61460, 61471, 61484, 61500 }; //C0~C7，270°
    { 61519, 61540, 61563, 61588, 61616, 61647, 61679, 61714 }; //C8~CF
    { 61752, 61791, 61833, 61877, 61924, 61972, 62023, 62075 }; //D0~D7
    { 62130, 62187, 62246, 62307, 62370, 62434, 62501, 62569 }; //D8~DF
    { 62640, 62712, 62785, 62861, 62938, 63016, 63096, 63177 }; //E0~E7
    { 63260, 63345, 63430, 63517, 63605, 63694, 63785, 63876 }; //E8~EF
    { 63969, 64062, 64156, 64251, 64347, 64444, 64541, 64639 }; //F0~F7
    { 64737, 64836, 64935, 65035, 65135, 65235, 65335, 65435 }; //F8~FF
```

11. Simulink 程式方塊參數設定說明

在本節中，將說明如何在 eMotion 系統的 DSP 程式發展環境中，為使用組合語言所開發的模組程式，建立相對應的 Simulink 程式方塊，使系統程式的聯結，與模組程式參數的調整，可以完全在 Matlab/Simulink 所提供的人機界面中完成。

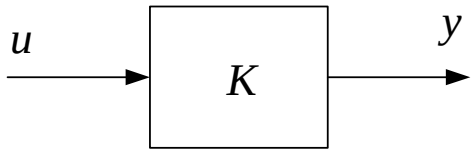
在 11.1 節中，將以完成增益值乘法的程式範例，介紹 Simulink 程式方塊的參數設定規則。接著在 11.2 節中，說明與 AC 馬達控制相關的模組程式，其 Simulink 程式方塊的參數設定值。

11.1 Simulink 程式方塊的參數設定範例

圖二十六所示，為實現增益值乘法的模組程式，計有一輸入，一輸出，與一可調整的增益值，*k*。該模組的組合語言如下：

Gain.asm 的程式列印：

```
@local  u     R0  IP;         //輸入點
@local  y     R1  OP;         //輸出點，乘上 k 值之後的結果
@local  k     R2  4096;       //功能方塊的增益值，1.0=4096
@data   3;                    //模組變數區長度
@initial:           //***** 開機啟動程序 *****
     WR  y,#0; RET;                              //y=0
@timer:             //***** 即時控制程序 *****
     RD (u);  MUL k;  SFR 12;  WR y;             //y =(u*k/4096)
     RET;
```



圖九 增益模組輸入輸出示意圖

程式中，選用 4096 代表實數系的 1.0，其主體在於乘法的實現(*MUL k*)與實數準位的調整(*SFR 12*)。詳細的組合語言定義，請參考 eMotion 系統的使用手冊。要為上述的程式，建立 Simulink 的程式方塊，其步驟相當簡單，詳述如下：

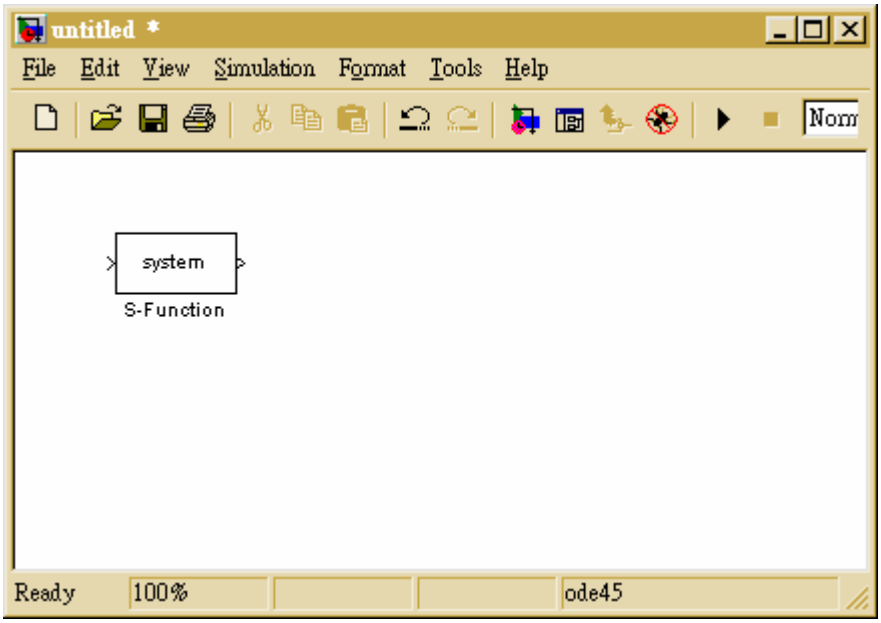
(1) 首先決定好模組程式的輸入、輸出、與內部變數的模組暫存器位置。

以 gain.asm 程式為例，輸入(IP)變數 u 的暫存器位址選定為 0，輸出(OP)變數 y

的暫存器位址為 1，內部變數 k 的暫存器位址為 2。

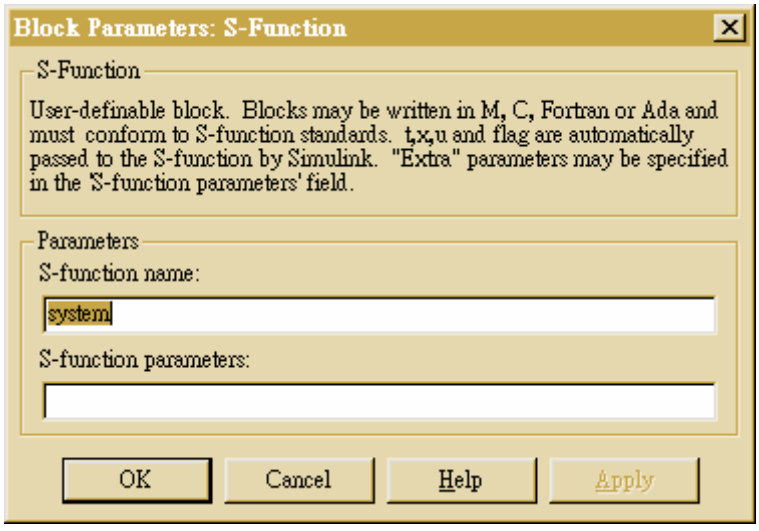
(2) 在 Simulink 中將 S-function 函式拖曳到新開啟的 Simulink 程式視窗，圖二十四。

Simulink → Funtions & Tables → S-function



圖十 在 Simulink 程式視窗中拉入 S-function 方塊

(3) 點選 S-function 方塊，出現以下的視窗畫面，圖二十五。


Block Parameters: S-Function
S-Function
User-definable block. Blocks may be written in M, C, Fortran or Ada and must conform to S-function standards. t,x,u and flag are automatically passed to the S-function by Simulink. "Extra" parameters may be specified in the 'S-function parameters' field.
Parameters
S-function name:
system
S-function parameters:
OK
Cancel
Help
Apply

圖十一 S-function 的參數輸入畫面

(4) 輸入變數的決定法則如下。

(4.1) S-function name 欄位，統一使用 **DspBlock** (注意大小寫)

DspBlock.dll 為完成聯結 Matlab/Simulink 與 eMotion，並進行資料傳遞的系統連線程式。

(4.2) 參數欄位，先輸入檔名。以本範例為例，輸入'gain'。

(4.3) 輸入欄位，預設的旗標為 30+輸入計數。以本範例而言，有一輸入，其旗標值為 30+1=31。另外，該輸入 u，所使用的暫存器編號為 0，整合在一起得，

[暫存器編號, 輸入旗標+輸入計數] = [0, 31]

(4.4) 輸出欄位，預設的旗標為 40+輸出計數。以本範例而言，有一輸出，其旗標值為 40+1=41。另外，該輸出 y，所使用的暫存器編號為 1，整合在一起得，

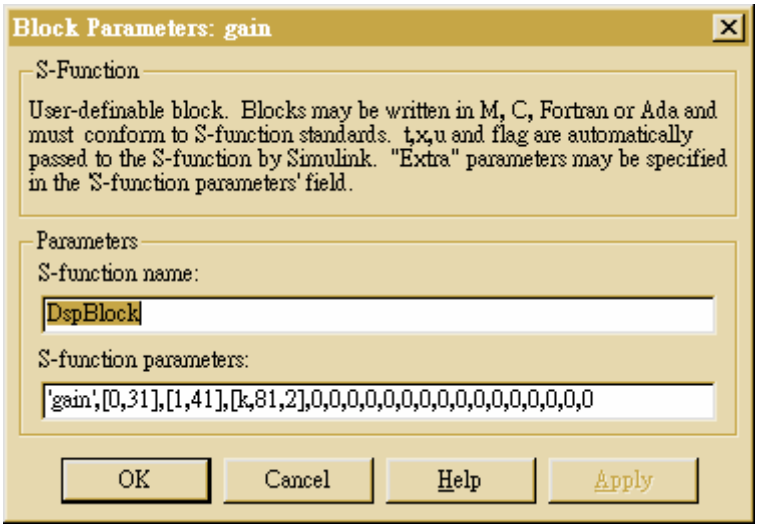
[暫存器編號, 輸出旗標+輸入計數] = [1, 41]

(4.5) 變數欄位，預設的旗標為 80+變數計數。以本範例而言，僅有一增益變數 k，所以旗標值為 80+1=81。另外，內部變數 k，所佔用的暫存器編號為 2，整合時依下列格式輸入，

[變數名, 變數旗標+編號, 暫存器編號] = [k, 81, 2]

(4.6) 最後，將參數值補 0 到總數為 20，中間以逗點分開。以本範例而言，將以上各參數加以整合，在 S-function 中的輸入值整合得，

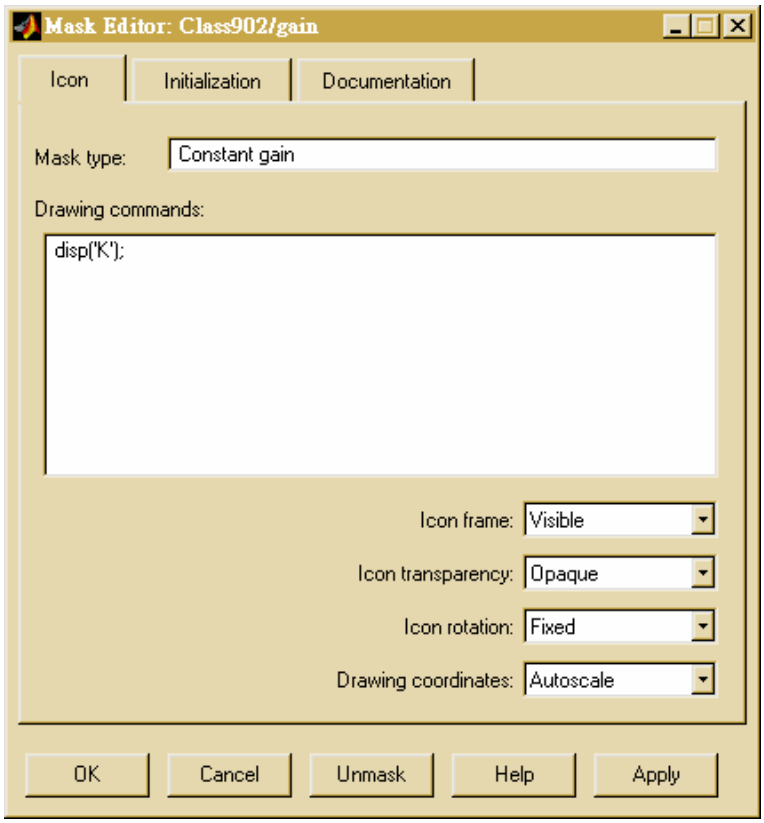
'gain', [0, 31], [1,41], [k, 81, 2], 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0



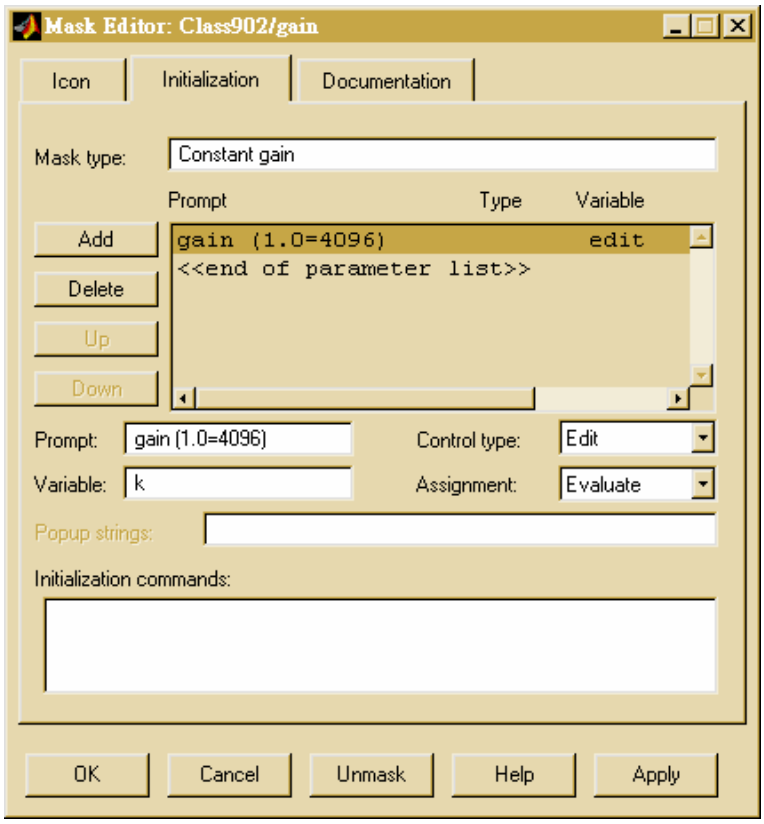
圖十二 輸入參數後的 S-function 畫面

(5) 首先點選程式方塊，再從表單中，選取選取 Edit → Edit Mask …
將出現圖二十七所示用來設定 Simulink 程式方塊的對話視窗，圖二十七。

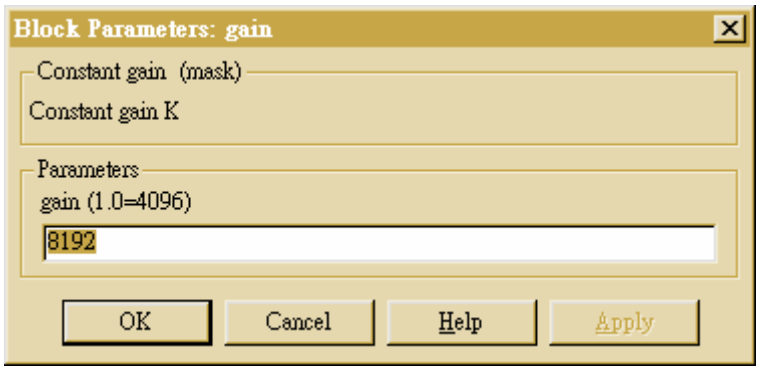
(6)依照視窗中，Icon → Initialization → Documentation 的順序，將參數填入。其中 Icon 表單，是用來對程式方塊的外表進行修飾。Initialization 表單，用來為程式方塊的參數與控制項，建立對話視窗，以本範例為例，為增益值 k，建立參數輸入的對話視窗，參見圖二十八與圖二十九。另外，Documentation 表單則為本程式方塊，建立註解，參見圖三十。以上的說明，僅是簡要的概述，欲知進階的參數使用方法，請參考 Matlab/Simulink 的使用手冊。



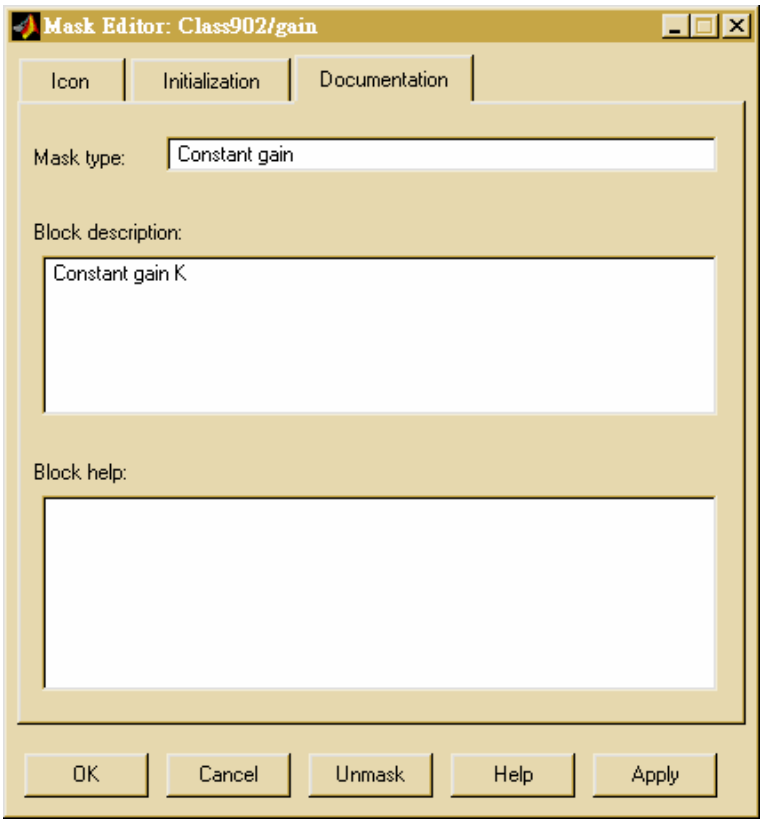
圖十三 Mask 程式方塊的對話視窗



圖十四 建立參數對話視窗的表單



圖十五 完成設定後的增益值輸入視窗

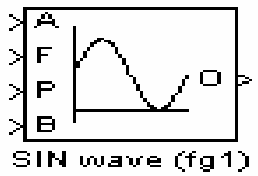


圖十六三 程式方塊註解設定表單

11.2 交流馬達控制方塊的參數設定說明

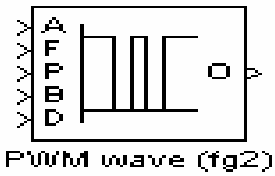
依據 11.1 節所說明的參數設規則，將用於控制交流馬達相關的 Simulink 程式方塊參數設定結果，綜整如下：
實驗中所使用的模組程式包括：

正弦波產生器：SIN_wave.asm



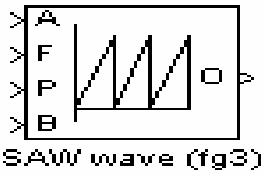
'SIN_wave',[0,31],[1,32],[2,33],[3,34],[4,41],[kAmp,81,7],[kFrq,82,8],[kTh,83,9],[kBias,84,10],0,0,0,0,0,0,0,0,0,0

方波產生器：PWM_wave.asm



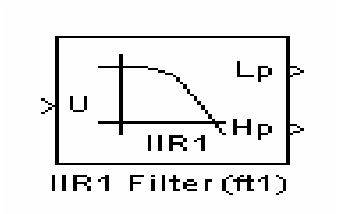
'PWM_WAVE',[0,31],[1,32],[2,33],[3,34],[4,35],[5,41],[kAmp,81,7],[kFrq,82,8],[kTh,83,9],[kBias,84,10],[kDuty,85,11],0,0,0,0,0,0,0,0

鋸齒波產生器：SAW_wave.asm



'SAW_WAVE',[0,31],[1,32],[2,33],[3,34],[4,41],[kAmp,81,6],[kFrq,82,7],[kTh,83,8],[kBias,84,9],0,0,0,0,0,0,0,0,0,0

IIR1_filter.asm　　：一階濾波器。



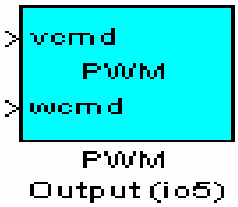
'IIR1_FILTER', [0, 31], [1, 41], [2, 42], [dt, 81, 3], [ka, 82, 4],0,0,0,0,0,0,0,0,0,0,0,0,0,0

DAS4.asm　　：資料蒐集模組。

ACM_protect.asm：驅動級的保護界面。

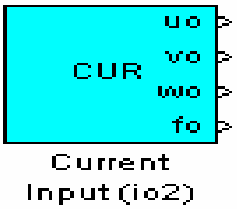
'ACM_protect',[0,31],[enb,81,1],[48+(pwm-1),82,2],[zone,83,3],0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

PWM_output.asm　：三相 PWM 驅動界面。



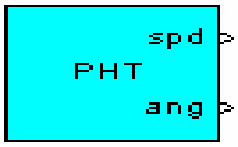
'PWM_outACM4',[0,31],[1,32],[80+3*(uout-1),81,3],[81+3*(vout-1),82,4],[82+3*(wout-1),83,5],[lmtH,84,6],0,0,0,0,0,0,0,0,0,0,0,0,0

CUR_input.asm　：三相電流回授界面。



'CUR_input',[0,41],[1,42],[2,43],[3,44],[72+(vi-1),81,4],[76+(wi-1),82,5],[vz,83,6],[wz,84,7],0,0,0,0,0,0,0,0,0,0,0

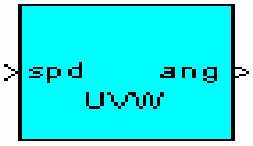
POS_input.asm　：位置和速度回授界面。



Position
Input(io3)

'POS_input',[0,41],[1,42],[104+2*(pht-1),81,3],0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

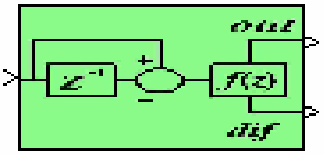
UVW_input.asm　：馬達相角回授界面。



Phase
Detect(io4)

'UVW_input',[0,31],[1,41],[96+(uvw-1),81,2],[kr,82,3],0,0,0,0,0,0,0,0,0,0,0,0,0,0

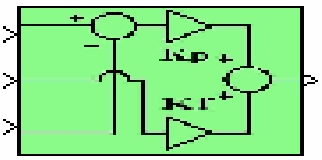
ACC_control.asm：加減速控制器。



ACC/DEC
Control(cn7)

'ACC_control',[0,31],[1,41],[2,42],[axis-1,81,3],0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

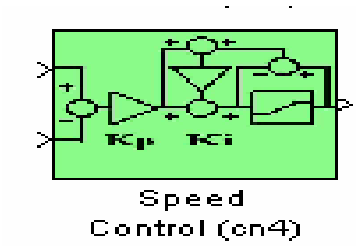
POS_control.asm：位置迴路控制器(包括前饋補償)。



Position
Control(cn6)

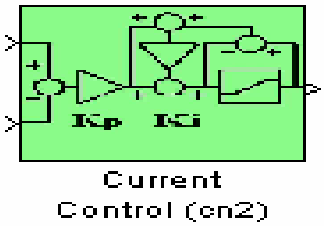
'POS_control',[0,31],[1,32],[2,33],[3,41],[axis-1,81,4],[kp,82,5],[kf,83,6],0,0,0,0,0,0,0,0,0,
0,0,0

SPD_control.asm：速度迴路控制器(PI 控制器)。



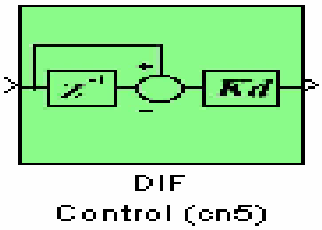
'SPD_control',[0,31],[1,32],[2,41],[axis-1,81,3],[kp,82,4],[ki,83,5],[high,84,6],[low,85,7],0,0,0,0,0,0,0,0,0,0,0

CUR_control.asm：電流迴路控制器(PI 控制器)。



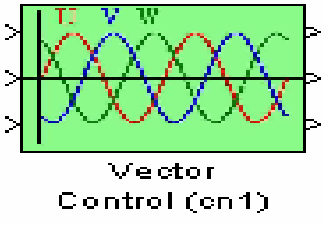
'CUR_control',[0,31],[1,32],[2,41],[kp,81,3],[ki,82,4],[high,83,5],[low,84,6],0,0,0,0,0,0,0,0,0,0,0,0

DIF_control.asm：差分控制器(速度回授計算)。



'DIF_control',[0,31],[1,41],[axis-1,81,2],[kd,82,3],0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

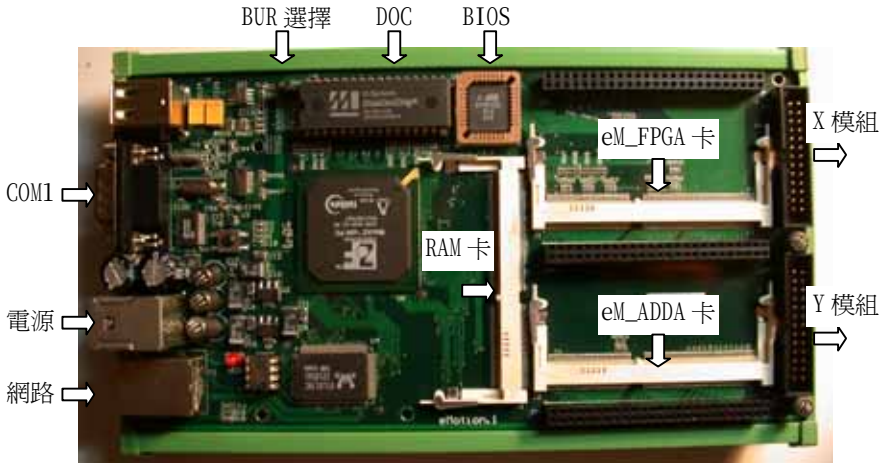
VEC_control.asm：相位分配控制器。



'VEC_control',[0,31],[1,32],[2,33],[3,41],[4,42],[5,43],[kTq,81,12],[kAng,82,13],[kBias,81,14],0,0,0,0,0,0,0,0,0,0

12. 系統安裝操作說明

Emotion 操作程序



測試重點：

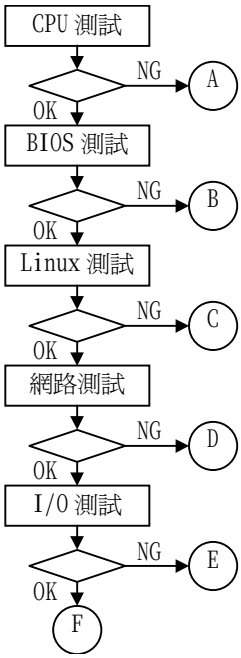
CPU測試　：ZFX86可以單獨測試CPU，只要用COM1接上PC即可顯示CPU訊息。

BIOS測試　：BIOS測試需要BIOS和RAM卡正常工作，就可從COM1顯示BIOS訊息。

Linux測試：一旦BIOS和DOC工作，Linux即可正常開機。

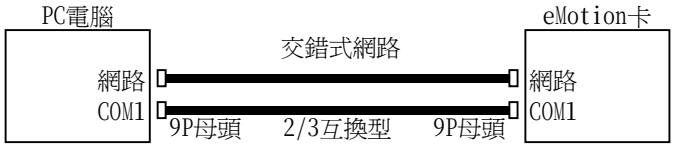
網路測試　：網路需要PCI-bus和網路IC都正常才能工作。

I/O測試　：經由FPGA控制I/O，即可確認全板工作正常。



測試的整體流程如下，其中的操作細節詳細說明如後：

PC 準備工作



1. PC必須準備COM1和網路連線，圖示如下：
   注意：因為eMotion卡和PC的COM1接頭定義完全一致，所以RS232接線必須採用2/3互換的接線型式，一般出廠已經過測試所以只準備交錯式網路線即可。
2. 將CD中的SMB子目錄全部抄到C碟中，建立
   C:\SMB
   的子目錄。並將所有檔案的唯讀設定取消，變成可讀寫的狀態。
3. 將C:\SMB目錄設為資源分享（注意smb必須為小寫）
   名稱：SMB，
   類型：完整，
   密碼：無。
   以後SMB的整個目錄就可透過網路讓eMotion卡分享。
4. 設定[控制台][網路]中的識別資料，固定
   電腦名稱：Jtek
   工作群組：WORKGROUP，並選擇
   IP位址：192.168.1.136
   網路遮罩：255.255.255.0
   WIN98系統直接在→控制台→網路→識別資料更改電腦名稱及工作組群，IP位址在網路卡TCP/IP內容設定IP位址與網路遮罩。
   WINXP及2000系統直接在→控制台→系統→電腦名稱更改電腦名稱及工作組群，IP位址在→控制台→網路連線→區域網路選擇內容→Internel Protocol（TCP/IP）內容設定IP位址與網路遮罩
   ，在2000系統必須建立使用者名稱及密碼，操作如下：控制台→使用者與密碼→選擇必須輸入使用者名稱與密碼→新增→eMotion→輸入密碼→選擇受限使用者→完成。
   ****注意：這些設定是可變的，不過就硬體測試用的電腦而言，請完全依照上面的設定，以統一測試條件，避免無謂的困擾。****
5. 基本上第4項所設定數值皆可更改，該設定初始值在DOC（硬碟IC）裡，更改方法在後面操作步驟說明，如PC已有網路卡並應用於其他使用（如區域網路或網際網路使用）為避免更改操作困難，建議加裝一片

網路卡，如USB網路卡其設定按照上面設定方式。

注意：安裝USB網路卡時無論插或拔一定要PC關機，否則電腦會受到嚴重當機，甚至必須重新安裝作業系統。

6. 將eMotion卡接上電源後約3分鐘

開機後用TelNet進入執行，程序如下：

A. 在視窗下執行 [開始][執行][telnet 192.168.1.2]

B. telnet視窗中的 [終端機][喜好設定] 中選擇

塊狀游標、VT100箭頭、VT100/ASCII

C. 進入Linux視窗中執行：

login: guest

password: guest

$ su

password: （無password）

cd /

7. 接下去可以繼續做網路連線，如您作業系統為 WIN98 則輸入 eMot98，如 WIN2000 則輸入 eMot2000

#./eMot98

```
eMotion:/initrd/root# cd /
eMotion:/# ./eMot98
Added interface ip=192.168.1.2 bcast=192.168.1.255
  nmask=255.255.255.0
Got a positive name query response from 192.168.1.136 (
  192.168.1.136 )
Initializing development environment. Please wait......
eMotion:/#
```

#./eMot2000 出現

password:輸入您設定密碼

＊ ＊＊如有連接不上時，請檢查以上步驟是否正確，如同時有 2 個以上網路卡時，請先關掉其他網路卡，等已正常連接後再開啟網路卡；假設您安裝防毒軟體，必須關掉防火牆，PC －cillin 2003 版本如已裝個人防火牆，請先移除後重新安裝，取消安裝個人防火牆；另外如電腦有安裝再生卡，必須移除。

＊ ＊＊如電腦名稱不要 Jtek，又不想改 DOC 裡資料，請不要執行 eMot98，先輸入 cat eMot98 出現

＊ smbmount //Jtek/smb /smb -N

按照以上指令輸入 smbmount //您目前電腦名稱 /smb /smb -N 即可。

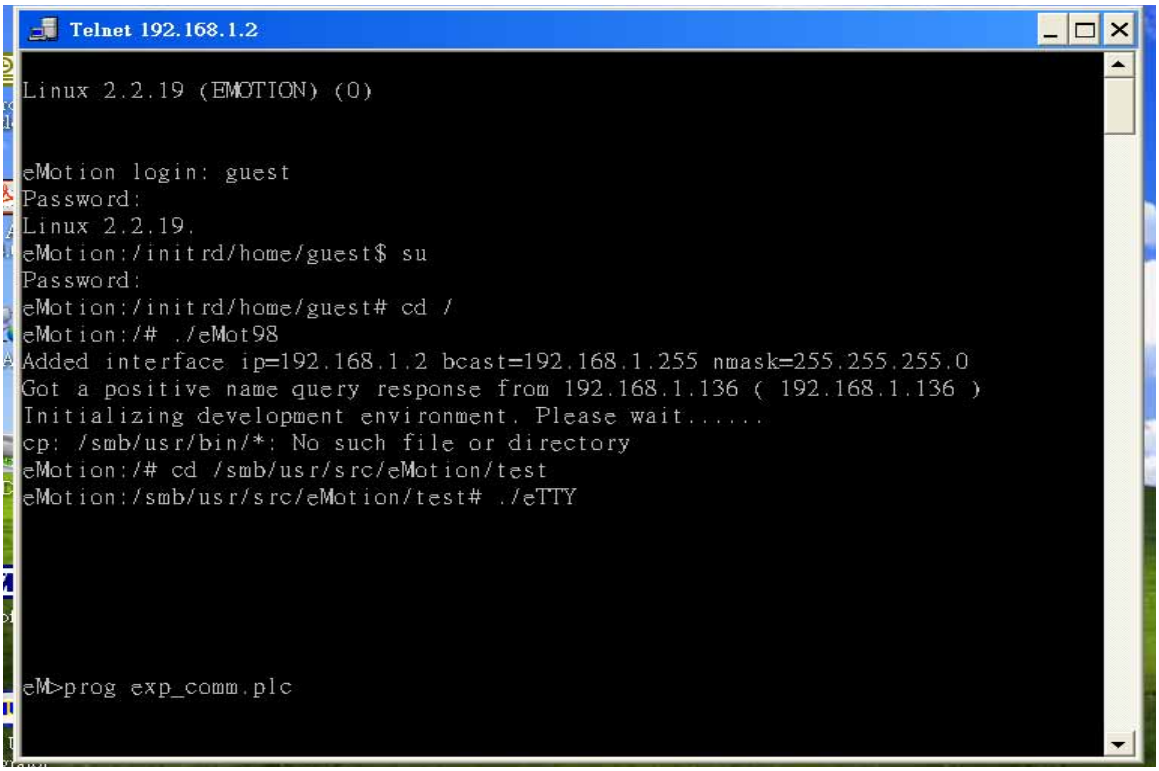
8、如欲執行數位通訊實驗時

#cd /smb/test

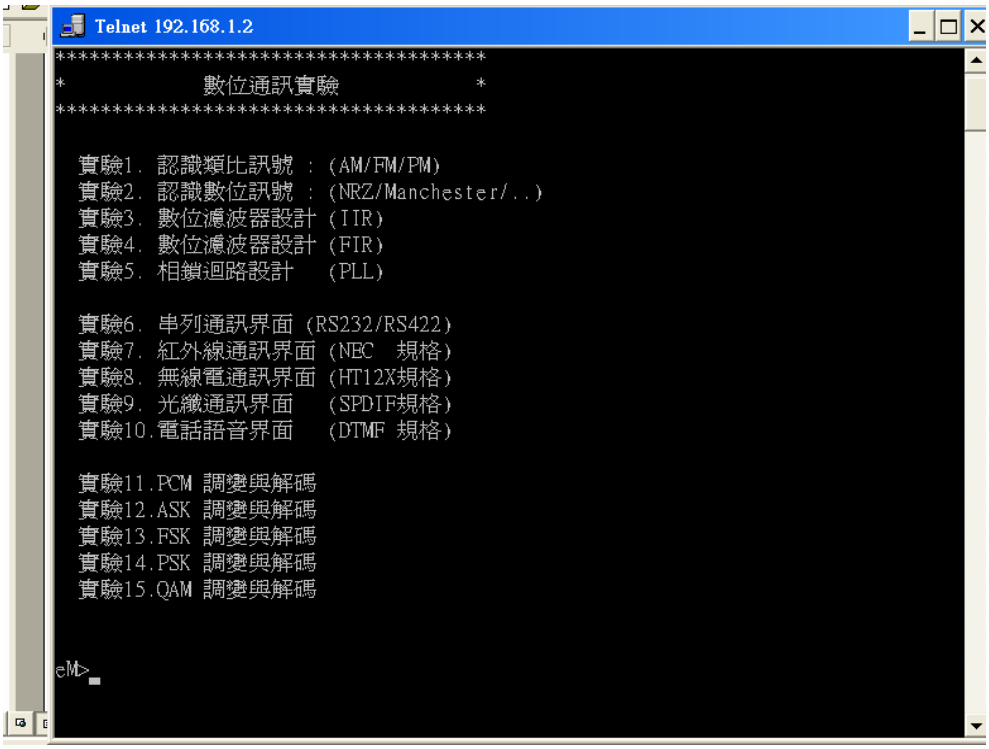
./eTTY

eM>prog exp_comm.plc

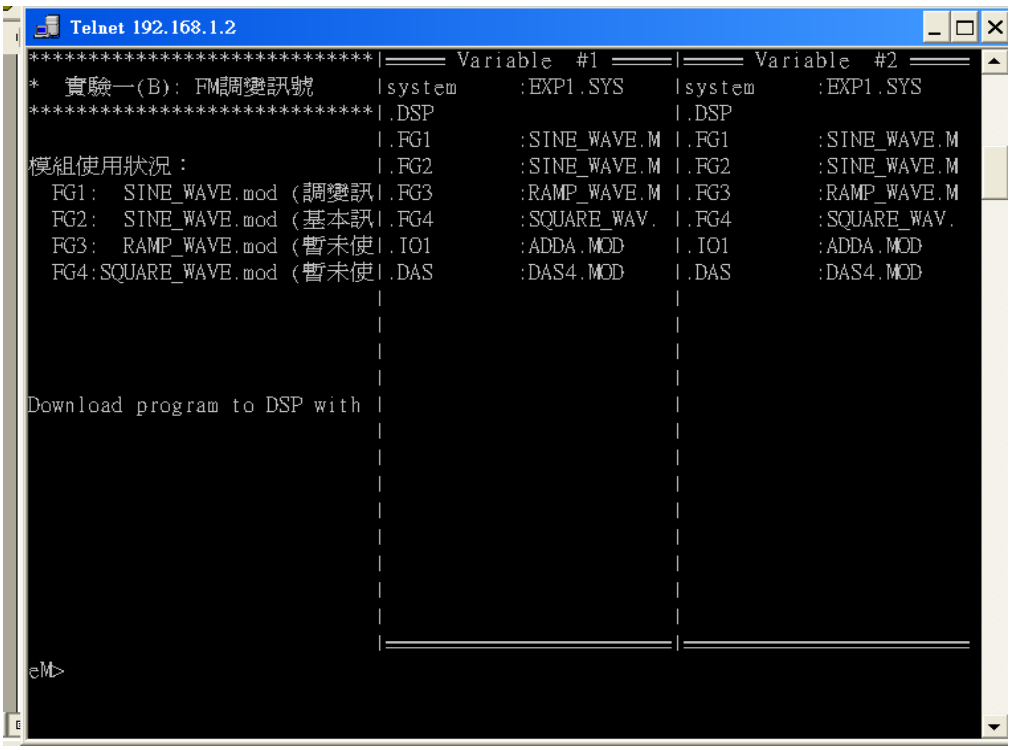
完整操作畫面顯示如下：



按照步驟選擇您要的實驗

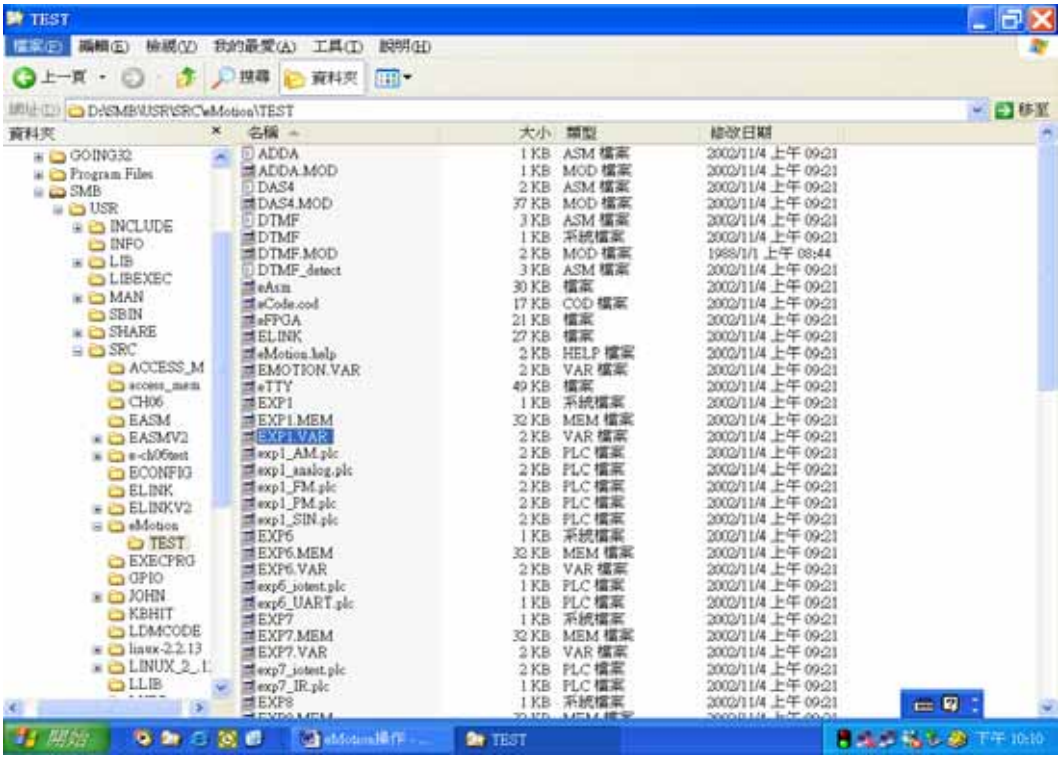


按 TAB 鍵右邊可顯示示波器及邏輯分析儀操作畫面
再按 TAB 鍵選擇 Variable#1, Variable#2 操作, ↑→↓←鍵設定參數值
F4 功能鍵設定顯示數值為 16、10 進位或正負值
ESC 結束示波器及邏輯分析儀操作



離開 TEST 目錄下按 Q 即可或輸入 exit 再輸入 exit 可離開 telnet 再關機

9、如要觀察下面所有檔案詳細內容請安裝 Textpad 軟體


TEST
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁 搜尋 資料夾
D:\SMB\USR\SRC\eMotion\TEST
資料夾
名稱
大小
類型
修改日期
GO!NG32
Program Files
SMB
USR
INCLUDE
INFO
LIB
LIBEXEC
MAN
SBIN
SHARE
SRC
ACCESS_M
access_mem
CH06
EASM
EASMV2
e-ch06test
ECONFIG
ELINK
ELINKV2
eMotion
TEST
EXECPRG
GPIO
JOHN
KBHIT
LDMCODE
linux-2.2.13
LINUX_2_1
LLIB
ADDA 1 KB ASM 檔案 2002/11/4 上午 09:21
ADDA.MOD 1 KB MOD 檔案 2002/11/4 上午 09:21
DAS4 2 KB ASM 檔案 2002/11/4 上午 09:21
DAS4.MOD 37 KB MOD 檔案 2002/11/4 上午 09:21
DTMF 3 KB ASM 檔案 2002/11/4 上午 09:21
DTMF 1 KB 系統檔案 2002/11/4 上午 09:21
DTMF.MOD 2 KB MOD 檔案 1988/1/1 上午 08:44
DTMF_detect 3 KB ASM 檔案 2002/11/4 上午 09:21
eAsm 30 KB 檔案 2002/11/4 上午 09:21
eCode.cod 17 KB COD 檔案 2002/11/4 上午 09:21
eFPGA 21 KB 檔案 2002/11/4 上午 09:21
ELINK 27 KB 檔案 2002/11/4 上午 09:21
eMotion.hlp 2 KB HELP 檔案 2002/11/4 上午 09:21
EMOTION.VAR 2 KB VAR 檔案 2002/11/4 上午 09:21
eTTY 49 KB 檔案 2002/11/4 上午 09:21
EXP1 1 KB 系統檔案 2002/11/4 上午 09:21
EXP1.MEM 32 KB MEM 檔案 2002/11/4 上午 09:21
EXP1.VAR 2 KB VAR 檔案 2002/11/4 上午 09:21
exp1_AM.plc 2 KB PLC 檔案 2002/11/4 上午 09:21
exp1_analog.plc 2 KB PLC 檔案 2002/11/4 上午 09:21
exp1_FM.plc 2 KB PLC 檔案 2002/11/4 上午 09:21
exp1_FM.plc 2 KB PLC 檔案 2002/11/4 上午 09:21
exp1_SIN.plc 2 KB PLC 檔案 2002/11/4 上午 09:21
EXP6 1 KB 系統檔案 2002/11/4 上午 09:21
EXP6.MEM 32 KB MEM 檔案 2002/11/4 上午 09:21
EXP6.VAR 2 KB VAR 檔案 2002/11/4 上午 09:21
exp6_jotest.plc 1 KB PLC 檔案 2002/11/4 上午 09:21
exp6_UART.plc 1 KB PLC 檔案 2002/11/4 上午 09:21
EXP7 1 KB 系統檔案 2002/11/4 上午 09:21
EXP7.MEM 32 KB MEM 檔案 2002/11/4 上午 09:21
EXP7.VAR 2 KB VAR 檔案 2002/11/4 上午 09:21
exp7_jotest.plc 2 KB PLC 檔案 2002/11/4 上午 09:21
exp7_IR.plc 1 KB PLC 檔案 2002/11/4 上午 09:21
EXP8 1 KB 系統檔案 2002/11/4 上午 09:21
開始
TEST
下午 10:10

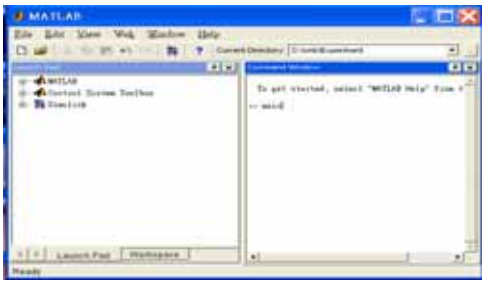
10、預使用 Matlab-Simulink（必須在 R6.1 版以上）操作時，

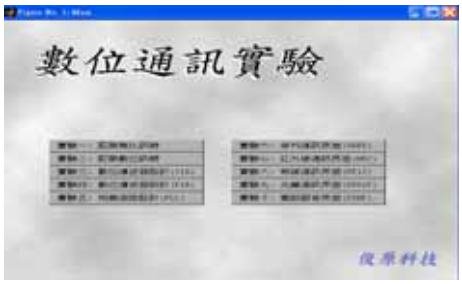
#cd /smb/simulink

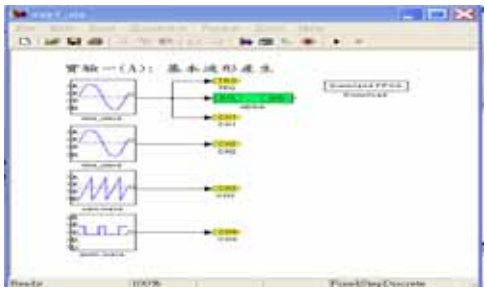
./eTTY 192.168.1.136

進入 Matlab 選擇路徑在 smb/ecomm

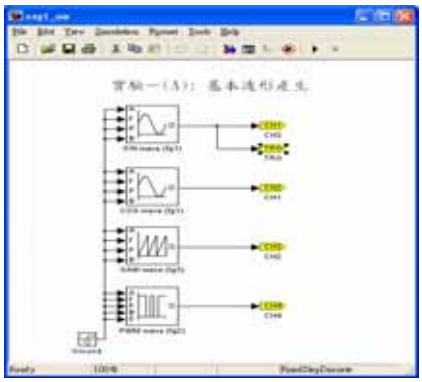
輸入 main 出現數位通訊實驗選擇

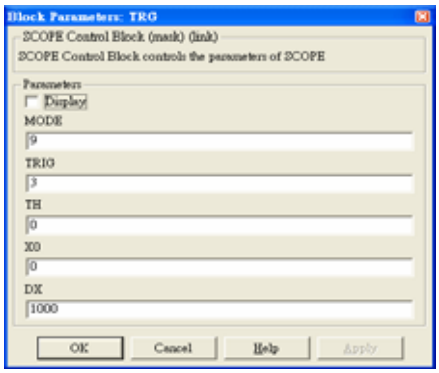


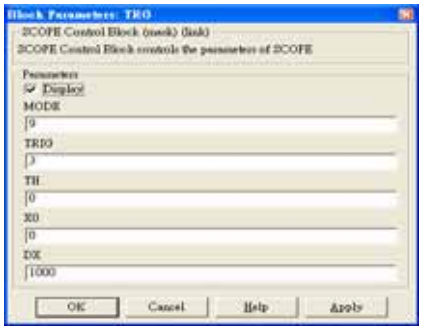


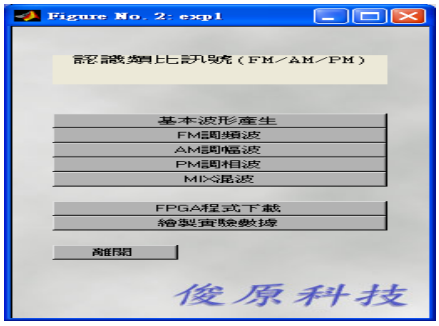


＊ ＊＊注意：smb/ecomm/emotion.dat 內容為
＊ c:\smb\simulink\
＊ smb/ecomm/plot_G.M 內容為
＊ c:\smb\simulink\
＊ 必須為您現在 SMB 位置 ，如不同請更改內容再存入。

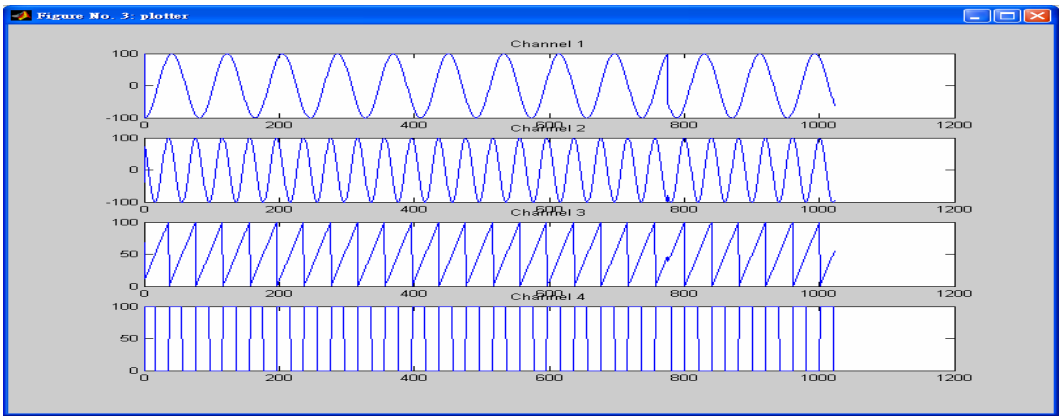








如要將現在示波器圖形在 simulink 顯示操作如下：
先將圖一：trg 打開，顯示如圖二：在 display 打 V 再按 apply 鍵，如圖三
再將在 display 取消打 V 再按 apply 鍵完成記錄，再按圖四：按下繪圖實驗數據，即可顯示下列圖形



按 TAB 鍵可顯示 LCD 示波器及邏輯分析儀右邊數字顯示

ESC 結束，才能恢復 simulink 操作環境

以上詳細資料可參考數位通訊使用手冊

11、開始選擇下載 FPGA 範例，以唐佩忠著 VHDL 數位邏輯設計為範例下面說明操作：(注意課本範例 UCF 配置檔必須全部更改)

詳細範例如下：

1. ch06.vhd, ch06.ucf, ch06.bit
   在 FPGA#2 中執行第六章範例程式，其中 ch06.vhd 完全未改。
2. ch06A.vhd, ch06A.ucf, ch06A.bit
   在 FPGA#2 中執行第六章範例程式，更改 vhd 程式使得 LED 閃亮頻率和範例接近。
3. ch06P250.vhd, ch06P250.ucf, ch06P250.bit
   在 FPGA#1 中執行第六章範例程式，改以 JP4 做輸出。所以並不方便顯示結果。
4. ch06P250A.vhd, ch06P250A.ucf, ch06P250A.bit
   在 FPGA#1 中執行第六章範例程式，改以 SIO 做輸出。所以必須搭配 FPGA#2 使用。
5. siotest1.vhd, siotest1.ucf, siotest1.bit
   搭配 ch06P250A 在 FPGA#2 中執行的轉換程式。
6. ch06P1K.bit
   和 ch06P250 一樣，只是必須在 2V1000 上執行。

7. ch06P1KA.bit
   和 ch06P250A 一樣，只是必須在 2V1000 上執行。

進入

#cd /smb/testFPGA

a. 下載 FPGA#1，鍵入：
   ./pgFPGA ch06P250.bit 1
b. 下載 FPGA#2，鍵入：
   ./pgFPGA ch06.bit 2

11、如欲執行 DC 伺服馬達控制實驗時

#cd /smb/testDCM
./eTTY
eM>prog exp_DCM.plc

預使用 Matlab-Simulink（必須在 R6.1 版以上）操作時，

#cd /smb/simulink
./eTTY 192.168.1.136
進入 Matlab 選擇路徑在 /smb/eDCM
輸入 exp_dcm 出現 DC 伺服馬達控制實驗選擇

12、如欲執行 AC 伺服馬達控制實驗時

#cd /smb/testACM
./eTTY
eM>prog exp_ACM.plc

預使用 Matlab-Simulink（必須在 R6.1 版以上）操作時，

#cd /smb/simulink
./eTTY 192.168.1.136
進入 Matlab 選擇路徑在 /smb/eACM8
輸入 exp_acm 出現 AC 伺服馬達控制實驗選擇

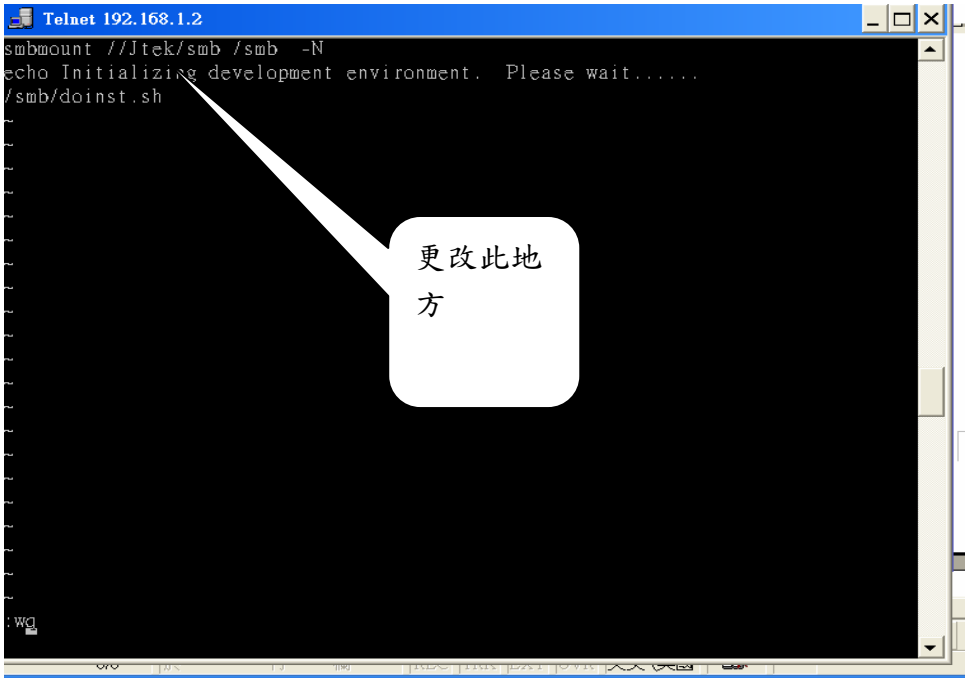
13、如欲執行 FPGA DSP 核心設計 VHDL 高等邏輯實驗時

#cd /smb/expDSP
./eTTY
eM>prog exp_DSP.plc

14、如想了解 Linux 基本操作指令，請看 SMB 裡檔案

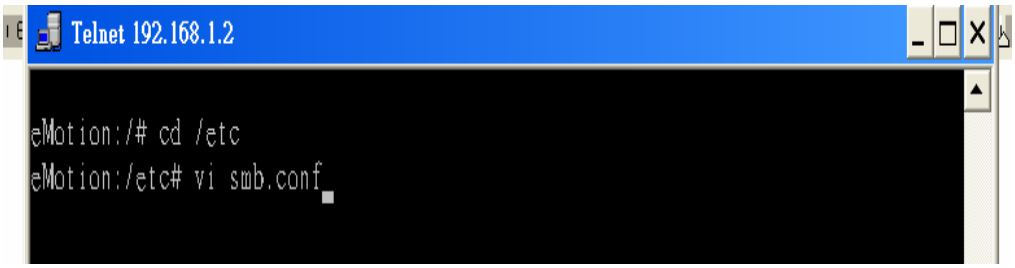
附註：如想更改電腦名稱（Jtek）或工作群組（WORKGROUP）時，
使用 VI 指令

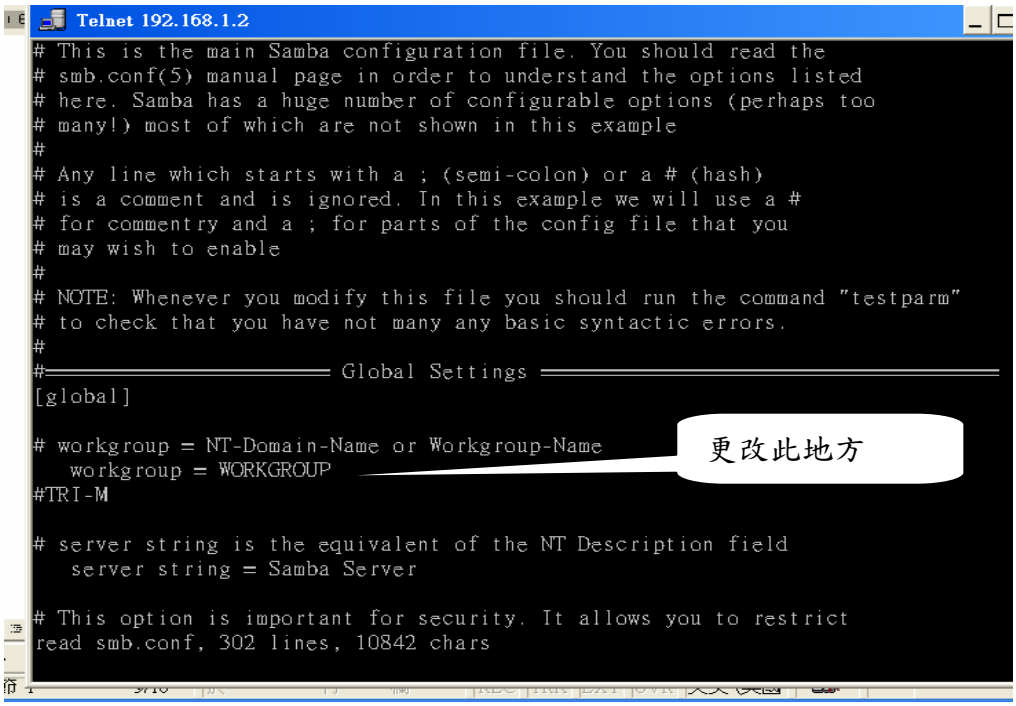
更改電腦名稱（Jtek）方法如下：



x→刪掉
i→插入
ESC→退回
：wq→儲存並結束

更改工作群組（WORKGROUP）方法：





最後再輸入 reboot　則寫入硬碟 IC（DOC）裡

注意：以上更改皆寫入硬碟 IC（DOC），更改原先設定，能不更改盡量不要更改。